Australia A $4.75 New Zealand NZ $6.50
Singapore S $9.45 Hong Kong H $23.50
Malaysia M $9.45 Sweden 30,-SEK

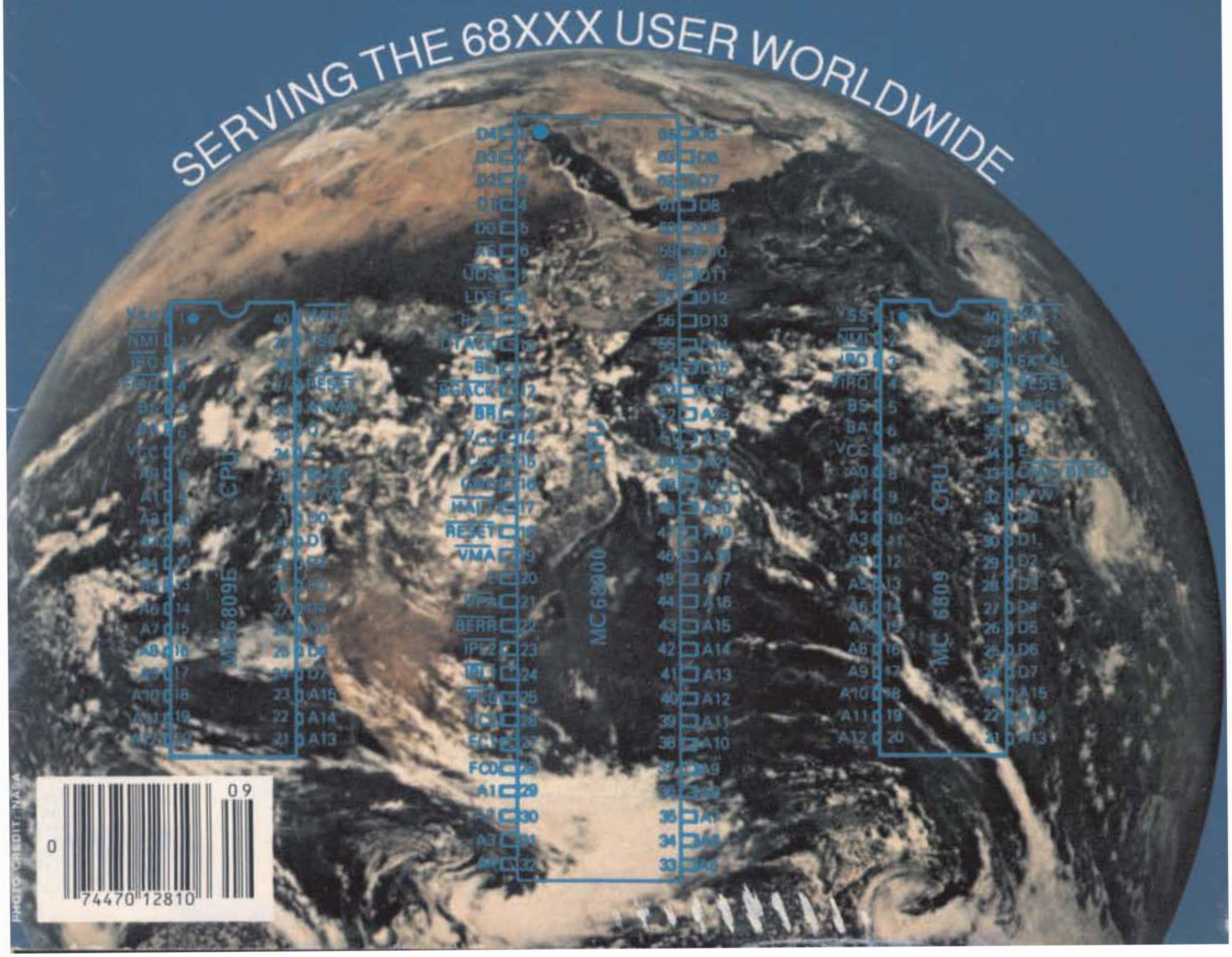
$2.95 USA

68' MICRO JOURNAL

Motorola VME-MACINTOSH-S 50 & Other 68XXX Systems
6809 68008 68000 68010 68020 68030
OS-9 The Magazine for Motorola CPU Devices FLEX SK*DOS
A User Contributor Journal

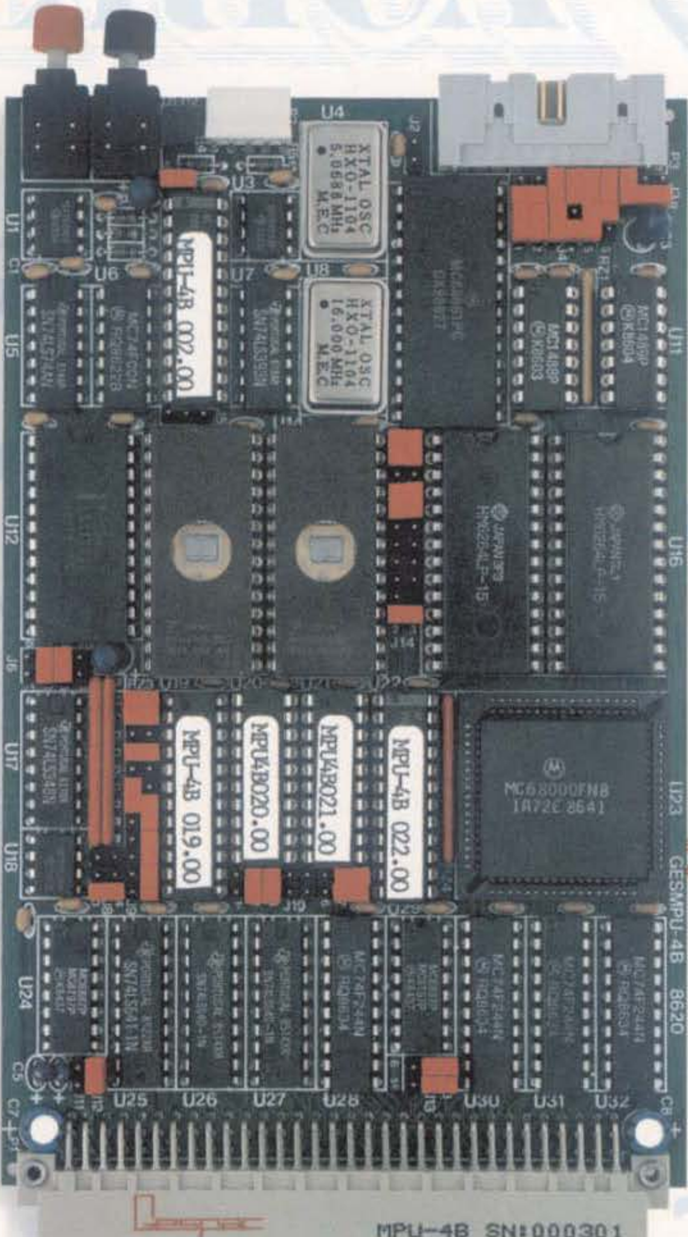
VOLUME IX ISSUE IX • Devoted to the 68XXX User • September 1987

The Grandfather of "DeskTop Publishing™"



0 74470 12810 09

GESPAC Gives You 68000 Performance at 8-bit Prices.



Actual Size

Introducing another Price / Performance breakthrough from GESPAC: A complete 68000 CPU module for $395 unit price. The GESMPU-48.

Wherever you thought of using an 8-bit microprocessor to do a job, you can now use the 16/32-bit GESMPU-48 instead. It will do the job, better, faster, and best-of-all for the same money.

On a single height Eurocard, we have packed an 8 MHz 68000 microprocessor (16 MHz optional), four sockets for up to 64 Kilobytes of zero-wait-states CMOS RAM and up to 128 Kilobytes of EPROM, one RS-232 serial port, and three 16-bit timers.

The GESMPU-48 is fully expandable through the standard G-64 bus, to accommodate up to 16 Megabytes of external memory. You can add any of more than 300 I/O modules available from GESPAC and a growing numbers of independent G-64 bus vendors.

To make your programming tasks easier, GESPAC supports the GESMPU-48 with the OS-9® and PDOS® real-time, multi-tasking operating systems and most popular high level languages and software development tools.

$395.00
Single Quantity

$316 @ 100 pieces

If you too like the idea of getting more for less, contact us today to receive information on the GESMPU-48 and the G-64 bus concept from GESPAC—the leader in single Eurocard microcomputer products worldwide.

Call Toll Free 1-800-4-GESPAC or Call (602) 962-5559.

IN USA - CANADA
50A West Hoover Ave.
Mesa, Arizona 85202
Tel. (602) 962-5559
Telex 386575

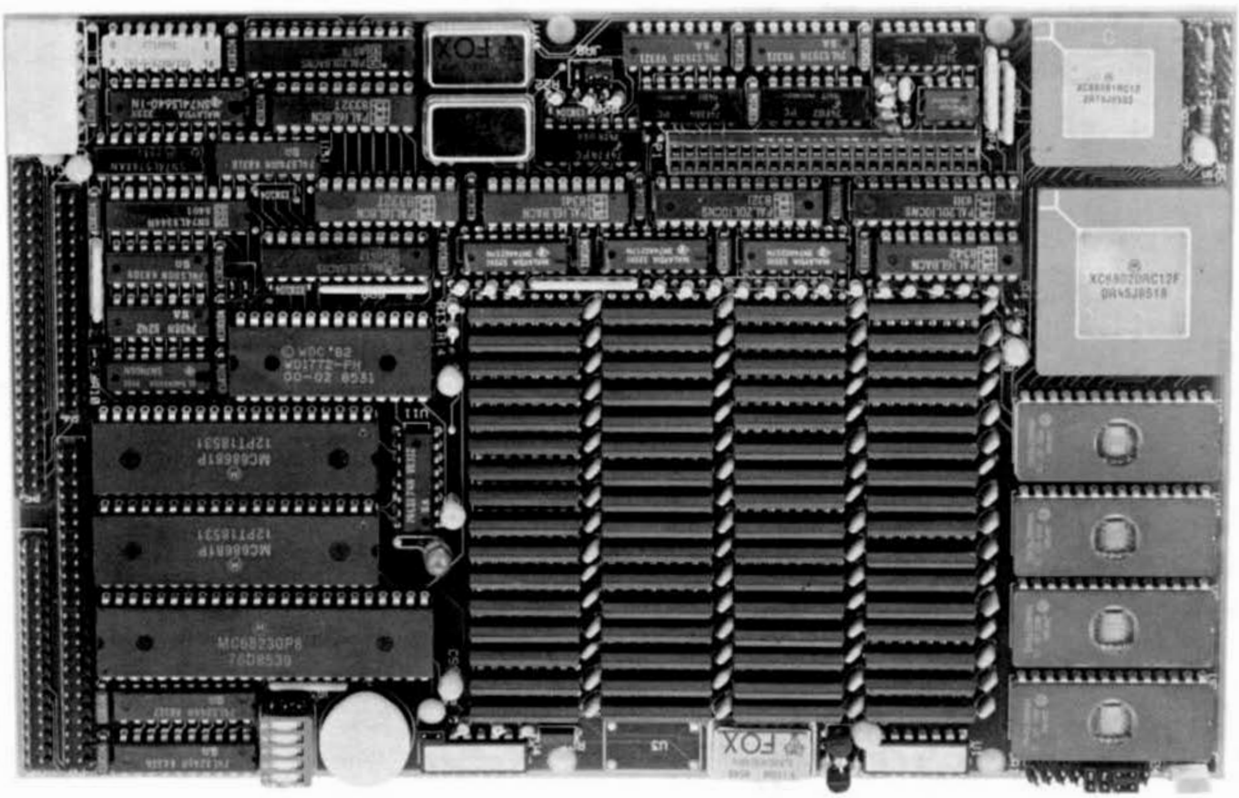
INTERNATIONAL
3, chemin des Aulx
CH-1228 Geneva
Tel. (022) 713400
Telex 429989

G-64 and G-96 are registered trademarks of GESPAC SA   OS-9 is a registered trademark of Microware Corp.   PDOS is a registered trademark of Eyring Research Institute Inc

GMX™ Micro-20
68020 SINGLE—BOARD COMPUTER

Mainframe CPU Performance on a 5.75″ x 8.8″ Board
(benchmark results available on request)

$2565.00 12.5 MHz Version
Quantity Discounts Available



Features

- 32-Bit MC68020 Processor (12.5, 16.67, or 20MHZ)
- MC68881 Floating-point coprocessor (optional)
- 2 Megabytes of 32-bit wide, high-speed RAM
- 4 RS-232 Serial I/O Ports (expandable to 36)
- 8-bit Parallel I/O Port ('Centronics' compatible)
- Time-of-Day Clock w/battery backup
- 16-bit I/O Expansion Bus
- Up to 256 Kbytes of 32-bit wide EPROM
- Floppy Disk Controller for two 5¼″ drives
- SASI Intelligent Peripheral Interface (SCSI subset)
- Mounts directly on a 5¼″ Disk Drive
- Optional Boards include Arcnet, Prototyping, I/O Bus adapter, 60 line Parallel I/O, RS-422/485

Software

Included:
- GMX Version of Motorola's 020Bug Debugger with up/download, breakpoint, trace, single-step, and assembler/disassembler capabilities
- Comprehensive Hardware Diagnostics

Optional:

UNIX™ like Multi-user/Multi-tasking Disk Operating Systems
- OS-9/68000™ (Real-time and PROMable)
- UniFLEX™

Programming Languages and Application Software
- BASIC, C, PASCAL, ABSOFT FORTRAN, COBOL and ASSEMBLER
- Spreadsheet, Data Base Management, and Word Processing

COMPLETE EVALUATION SYSTEMS AVAILABLE

GMX inc 1337 W. 37th Place Chicago, IL 60609

(312) 927-5510 • TWX 910-221-4055
State-of-the-Art Computers Since 1975

A Member of the CPI Family

68 Micro Journal

10 Years of Dedication to Motorola CPU Users

6800 6809 68000 68010 68020

The Originator of "DeskTop Publishing™"

Publisher
Don Williams Sr.

Executive Editor
Larry Williams

Production Manager
Tom Williams

Office Manager
Joyce Williams

Subscriptions
Kristi Hart

Contributing & Associate Editors

Ron Anderson — Dr. E.M. "Bud" Pass
Ron Voights — Art Weller
Doug Lurie — Dr. Theo Elbert
Ed Law — & Hundreds More of Us

Contents

68 MICRO JOURNAL

"Contribute Nothing - Expect Nothing" DMW 1986

COMPUTER PUBLISHING, INC.

"Over a Decade of Service"

"World Wide"

68 MICRO JOURNAL
Computer Publishing Center
5900 Cassandra Smith Road
PO Box 849
Hixson, TN 37343
Phone (615) 842-4600 Telex 510 600-6630

Copyrighted © 1987 by Computer Publishing, Inc.

68 Micro Journal is the *original* "DeskTop Publishing" product and has continously published s nce 1978 using only micro-computers and special "DeskTop" software. Using first a kit built 6800 micro-computer, a modified "ball" typewriter, and "home grown" DeskTop Publishing software. None was commercially available at that time. For over 10 years we have been doing "DeskTop Publishing"! *We originated what has become traditional "DeskTop Publishing"!* Today 68 Micro Journal is acknowledged as the "Grandfather" of "DeskTop Publishing" technology.

---

68 Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage paid ISSN 0194-5025 at Hixson, TN. and additional entries. Postmaster: send form 3597 to 68 Micro Journal, POB 849, Hixson, TN 37343.

Subscription Rates

1 Year $24.50 USA, Canada & Mexico $34.00 a year. Others add $12.00 a year surface, $48.00 a year Airmail, USA funds. 2 years $42.50, 3 years $64.50 plus additional postage for each additional year.

Items or Articles for Publication

Articles submitted for publication must include authors name, address, telephone number, date and a statement that the material is original and the property of the author. Articles subm tted should be on d skeue OS-9, SK*DOS, FLEX, Macintosh or MS-DOS. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please! Diagrams o.k.

Please - do not format with spaces any text indents, charts, etc. (source listing o.k.). We will edit in all formatting. Text should fall flush left and use d carriage return only to indicate a paragraph end. Please write for free authors guide.

Letters & Advertising Copy

Letters to the Editor should be the original copy, signed! Letters of grip as well as praise are acceptable. *We reserve the right to reject any letter or advertising material, for any reason we deem advisable.* Advertising Rates: Commercial please contact 68 Micro Journal Advertising Department. Classified advertising must be non-commercial. Minimum of $15.50 for first 15 words. Add $.60 per word thereafter. No classifieds accepted by telephone.

EXCITING SOFTWARE FROM THE LEADER...

microware

OS-9 ELECTRONIC MAIL

Flash your message on Electronic Mail Mail is a screen- or line-oriented program that runs on your OS-9/680X0 systems or over OS-9/NET. You can use distributed mailing lists or consecutive mailing list to get your message delivered. And received mail can be sent directly to your printer for immediate printout, spooled on a multiuser system or saved to a file. Mail features on-line help and complete, easy to understand documentation.
Electronic Mail $150.00.

PRINT SPOOLER

Spool it and Print it! Someone beat you to the printer? Don't blow your top while you cool your heels--get the OS-9/68000 Print Spooler and relax. The full-featured Print Spooler automatically routes and monitors the status of your devices and the output files to be spooled. Now you can have a complete print spooling management system at an affordable price.
OS-9 Print Spooler $150.00

FORTRAN

Crunch It! with Our New FORTRAN 77 Compiler Now you have a powerful new tool to take full advantage of the 68000 family of microprocessors. With Microware's FORTRAN 77 Compiler you can generate code that uses system-wide modules instead of linking redundant copies of the standard library to each program. Result: less memory, less disk space, faster loading and external updating!
FORTRAN 77 Compiler $750.00

OS-9/ST

NEW for your Atari ST! Now you can have the power of OS-9 on your Atari 520 or 1040 ST. A true multi-tasking environment for professional real-time results. OS-9/ST is available in two configurations: Personal and Professional. Choose either version for true multi-user support. And all at a price that puts UNIX to shame.

Personal OS-9/ST combines the power of OS-9 with an interactive, structured Basic. **$150.00**

Professional OS-9/ST has a powerful Assembler, Linker and User Debugger and the tools to turn your Atari ST into a full C Language workstation. **$600.00**

OS-9/68020 C COMPILER

NEW "speed demon" C Compiler! Now you can get your hands on a highly optimized C Language power tool--the OS-9/68020 C Compiler. When coupled with the MC68881 math co-processor, this compiler will let you'll blast through complex math functions in the blink of an eye. All compiler/assembler/linker options are controlled by an intelligent compiler executive that spares you from memorizing compiler options and module-calling sequences. And the compiler includes library functions for memory management and system events, and much, much more! The new OS-9/68020 C Compiler is included with the Professional OS-9/68020 System Software Package.
New C Language Compiler $750.00

To order these exciting NEW products or for more information... ***CALL TODAY!***

Microware Systems Corporation
1900 N.W. 114th Street • Des Moines, Iowa 50322
Phone 515-224-1929 • Telex 910-520-2535

West Coast Office
4401 Great American Parkway • Suite 220
Santa Clara, California 95054

Microware Japan, Ltd.
41-19 Honcho 4-Chome, Funabashi City • Chiba 273, Japan • Phone 0473 (28) 4493 • Telex 781-299-3122

Micromaster Scandinavian AB
S:t Persgatan 7
Box 1309
S751-43 Uppsala
Sweden
Phone: 018-138595
Telex: 76129

Dr. Rudolf Keil, GmbH
Porphystrasse 15
D-6905 Schriesheim
West Germany
Phone: (0 62 03) 67 41
Telex: 465025

Elsoft AG
Zelweg 12
CH-5405 Baden-Dattwil
Switzerland
Phone: (056) 83-3377
Telex: 828275

Vivaway Ltd.
36-38 John Street
Luton, Bedfordshire, LU1 2JE
United Kingdom
Phone: (0582) 423425
Telex: 825115

Microprocessor Consultants
92 Bynya Road
Palm Beach 2108
NSW Australia
Phone: 02-919-4917

Microdata Soft
97 bis, rue de Colombes
92400 Courbevoie
France
Phone: 1-788-80-80
Telex: 615405

Microware is on the move. We have openings for Technical and Marketing Professionals.
Send your resume (in confidence) today and find out more about these exciting opportunities.

OS-9 and BASIC09 are trademarks of Microware and Motorola. UNIX is a trademark of Bell Laboratories, Inc.

MUSTANG-020 Super SBC ™

DATA-COMP proudly presents the first Under $5000 "SUPER MICRO".

The MUSTANG-020™

CPI
Mustang-020™
A DATA-COMP
Hi-Speed
Product

MUSTANG-020™

The MUSTANG-020 68020 SBC provides a powerful, compact, 32 bit computer system featuring the "state of the art" Motorola 68020 "super" micro-processor. It comes standard with 2 megabyte of high-speed SIP dynamic RAM, serial and parallel ports, floppy disk controller, a SASI hard disk interface for intelligent hard disk controllers and a battery backed-up time-of-day clock. Provisions are made for the super powerful Motorola MC68881 floating point math co-processor, for heavy math and number crunching applications. An optional network interface uses one serial (four (4) standard, expandable to 20) as a 125/bit per second network channel. Supports as many as 32 nodes.

The MUSTANG-020 is ideally suited to a wide variety of applications. It provides a cost effective alternative to the other MC68020 systems now available. It is an excellent introductory tool to the world of hi-power, hi-speed new generation "super micros". In practical applications it has numerous applications, ranging from scientific to education. It is already being used by government agencies, labs, universities, business and practically every other critical applications center, worldwide, where true multi-user, multi-tasking needs exist. The MUSTANG-020 is UNIX C level V compatible. Where low cost and power is a must, the MUSTANG-020 is the answer, as many have discovered. Proving that price is not the standard for quality!

As a software development station, a general purpose scientific or small to medium business computer, or a super efficient real-time controller in process control, the MUSTANG-020 is the cost effective choice. With the optional MC68881 floating point math co-processor installed, it has the capability of systems costing many times over it's total acquisition cost.

With the DATA-COMP "total package", consisting of a heavy duty metal cabinet, switching power supply with rf/line by-passing, 5 inch DS/DD 80 track floppy, Xebec hard disk controller, 25 megabyte winchester hard disk, four serial RS-232 ports and a UNIX C level V compatible multi-tasking, multi-user operating system, the price is under $5000, w/12.5 megahertz system clock (limited time offer). Most all popular high level languages are available at very reasonable cost. The system is expandable to 20 serial ports, at a cost of less than $65 per port, in multiples of 8 port expansion options.

The system SBC fully populated, quality tested, with 4 serial ports pre-wired and board mounted is available for less that $3000. Quantity discounts are available for OEM and special applications, in quantity. All that is required to bring to complete "system" standards is a cabinet, power supply, disks and operating system. All these are available as separate items from DATA-COMP.

A special version of the Motorola 020-BUG is installed on each board. 020-BUG is a ROM based bebugger package with facilities for downloading and executing user programs from a host system. It includes commands for display and modification of memory, breakpoint capabilities, a powerful assembler/disassemble and numerous system diagnostics. Various 020-BUG system routines, such as I/O handlers are available for user programs.

Nonnal system speed is 3-4.5 MIPS, with burst up to 10 MIPS, at 16.6 megahertz. Intelligent I/O available for some operating systems.

Hands-on "actual experience sessions", before you buy, are available from DATA-COMP. Call or write for additional information or pricing.

DATA-COMP
Installed Systems World-Wide
OVER [illegible] YEARS OF DEDICATED QUALITY!

CPI
A Division of
Computer Publishing, Inc.
[illegible]
Hixson, Tn 37343
Telephone 615 842-4600
Telex 510 600-6630

Mustang-020 Mustang-08 Benchmarks

	32 bit Integer	Register Long
IBM AT 7300 Xenix Sys 3	9.7	
AT&T 7300 UNIX PC 68010	7.2	4.3
DEC VAX 11/780 UNIX Berkley 4.2	3.6	3.2
DEC VAX 11/750 " "	5.1	3.2
68008 OS-9 68K 8 Mhz	18.0	9.0
68000 OS-9 68K 10 Mhz	6.5	4.0
MUSTANG-08 68008 OS-9 68K 10 Mhz	***9.8***	***6.3***
MUSTANG-020 68020 OS-9 68K 16 Mhz	***2.2***	***0.88***
MUSTANG-020 68020 MC68881 UniFLEX 16 Mhz	***1.8***	***1.22***

```
Main()
    {
        register long i;
        for (i=0; i < 999999; ++i);
    }
```

Estimated MIPS - MUSTANG-020 ..... 4.5 MIPS,
Burst to 8 - 10 MIPS: Motorola Specs

OS-9

OS-9 Professional Ver	$850.00
*Includes C Compiler	
Basic09	500.00
C Compiler	500.00
68000 Disassembler (w/source add: $100.00)	100.00
Fortran 77	750.00
Microware Pascal	500.00
Omegasoft Pascal	900.00
Stylo-Graph	495.00
Stylo-Spell	195.00
Stylo-Merge	175.00
Stylo-Graph-Spell-Merge	695.00
PAT w/C source	229.00
JUS w/C source	79.95
PAT/JUST Combo	249.50
Sculptor+ (see below)	995.00
COM	125.00

UniFLEX

UniFLEX (68020 ver)	$450.00
Screen Editor	150.00
Sort-Merge	200.00
BASIC/PreCompiler	300.00
C Compiler	350.00
COBOL	750.00
CMODEM w/source	100.00
TMODEM w/source	100.00
X-TALK (see Ad)	99.95
Cross Assembler	50.00
Fortran 77	450.00
Sculptor+ (see below)	995.00

Standard MUSTANG-020™ shipped 12.5 Mhz.	
Add for 16.6 Mhz 68020	373.00
Add for 16.6 Mhz 68881	375.00
Add for 20 Mhz 68020/RAM	750.00
16 Port exp. RS-232 requires 1 or 2 Adapter Cards below.	335.00
RS232 Adapter Each card supports 4 additional ser. ports (total of 36 serial ports supported)	165.00
60 line Parallel I/O card Uses 3 68230 Interface/Timer chips, 6 groups of 8 lines each, separate buffer direction control for each group.	398.00
Prototype Board areas for both dip and PGA devices & a pre-wired memory area up to 512K DRAM.	75.00
SBC-AN Interface between the system and ARCNET modified token-passing LAN, fiber optics optional - call	475.00
LAN software drivers	120.00
Expansion for Motorola I/O Channel Modules	$195.00

Special for complete MUSTANG-020™ system buyers - Sculptor+ $695.00. SAVE $300.00

Software Discounts

All MUSTANG-020™ system and board buyers are entitled to discounts on all listed software: 10-70% depending on item. Call or write for quotes. Discounts apply after the sale as well.

THE PRO!

Only the "PRO" version of OS-9 supported!

This is *HEAVY DUTY* Country!

UPGRADES
Write or Call for Professional OS-9 "Full Bore" Upgrade Kit

For a limited time we will offer a $400 trade-in on your old 68XXX SBC. Must be working properly and complete with all software, cables and documentation. Call for more information.

- 12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path
- 32-bit wide data and address buses, non-multiplexed
- on chip instruction cache
- object code compatible with all 68XXX family processors
- enhanced instruction set - math co-processor interface
- 68881 math hi-speed floating point co-processor (optional)
- direct extension of full 68020 instruction set
- full support IEEE P754, draft 10.0
- transcendental and other scientific math functions
- 2 Megabyte of SIP RAM (512 x 32 bit organization)
- up to 256K bytes of EPROM (64 x 32 bits)
- 4 Asynchronous serial I/O ports standard
- optional to 20 serial ports
- standard RS-232 interface
- optional network interface
- buffered 8 bit parallel port (1/2 MC68230)
- Centronics type pinout
- expansion connector for I/O devices
- 16 bit data path
- 256 byte address space
- 2 interrupt inputs
- clock and control signals
- Motorola I/O Channel Modules
- time of day clock/calendar w/battery backup
- controller for 2, 5 1/4" floppy disk drives
- single or double side, single or double density
- 35 to 80 track selectable (48-96 TPI)
- SASI interface
- programmable periodic interrupt generator
- interrupt rate from micro-seconds to seconds
- highly accurate time base (5 PPM)
- 5 bit sense switch, readable by the CPU
- Hardware single-step capability

Don't be mislead! ONLY Data-Comp delivers the Super MUSTANG-020

These *hi-speed* 68020 systems are presently working at *NASA, Atomic Energy Commission, Government Agencies as well as Universities, Business, Labs, and other Critical Applications Centers*, worldwide, where speed, math crunching and *multi-user, multi-tasking UNIX C level V* compatability and low cost is a must!

MUSTANG-020 SBC		$2490.00
Cabinet w/switching PS		$299.95
5"-80 track floppy	DS/DD	$269.95
Floppy cable		$39.95
OS-9 68K Professional Ver. * Includes C Compiler ($500.00)		$850.00
Winchester cable		$39.95
Winchester Drive 25 Mbyte		$895.00
Hard Disk controller		$395.00
Shipping USA UPS		$20.00
Total:	Save $1000.00	$5,299.80

complete sytem $4,299.80

UniFLEX	Less	$100.00
MC68881 f/p math processor	Add	$275.00
16.67 Mhz MC68020		$375.00
16.67 Mhz MC68881		$875.00
20 Mhz MC68020 Sys		$750.00

Note all 68881 chips work with 20 Mhz Sys

NOTE: Only Professional OS-9 now available (68020 Version) Includes ($500.00) C Compiler 68020 & 68881 supported

complete
25 Mbyte HD System $4299.80
85 Mbyte HD System $5748.80

Data-Comp Division
A Decade of Quality Service"
Systems World-Wide
Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 · Telex 510 600-6630 Hixson, Tn 37343

PAT - JUST

PAT

With 'C' Source

$229.00

All OS-9 68XXX Systems

For Over a Decade!
1,200 Titles +

A Division of CPI

S.E. Media
5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

6800-6809-68008-68000-68010-68020

PAT FROM S. E. MEDIA -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved **PIE**, this is for **YOU! All PIE features & much more!** Too many features to list. And if you don't like ours, change or add your own. **C source included.** Easily configured to your CRT terminal, with special configuration section. No sweat!

68008 - 68000 - 68010 - 68020 OS-9 68K $229.00

COMBO PAT/JUST

Special $249.00

JUST

JUST from **S. E. MEDIA - -** Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The ONLY stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

68008 - 68000 - 68010 - 68020 OS-9 68K
With 'C' source $79.95

An Ace of a System in Spades! The New

MUSTANG-08/A™

Now with 4 serial ports standard & speed increase to 12 Mhz CPU + on board battery backup and includes the PROFESSIONAL OS-9 package - including the $500.00 OS-9 C compiler! This offer won't last forever!

NOT 128K, *NOT* 512K
FULL 768K No Wait RAM

The MUSTANG-08™ ystem took every hand from all other 68008 sysems we tested, running OS-9 68K!

The MUSTANG-08 includes OS9-68K™ and/or Peter Stark's SK*DOS. SK*DOS is a ingle user, single tasking system that takes up where *FLEX™ left off. SK*DOS is actually a 68XXX FLEX type system *(Not a TSC product)*

The OS-9 68K system is a full blown multi-user, multi-tasking 68XXX system. All the popular 68000 OS-9 software runs. It is a *speed whiz on disk I/O*. Fact is the MUSTANG-08 is faster on disk access than some other 68XXX systems are on memory cache access. Now, that is fast! And that is just a small part of the story! **See benchmarks.**

System includes OS-9 68K or SK*DOS - Your Choice

Specifications:

CPU	MC68008	12 Mhz
RAM	768K	256K Chips
	No Wait States	
PORTS	4 - RS232	MC68681 DUART
	2 - 8 bit Parallel	MC6821 PIA
CLOCK	MK48T02	Real Time Clock Bat. B/U
EPROM	16K, 32K or 64K	Selectable
FLOPPY	WD1772	5 1/4 Drives
HARD DISK	Interface Port	WD1002 Board

Now more serial ports - faster CPU Battery B/U - and $850.00 OS-9 Professional with C compiler included!

***$400.00** See Mustang-02 Ad - page 5 for trade-in details

MUSTANG-08

Mustang Hi-Speed Systems Only from Data-Comp Div. 68008-68030

LOOK

Seconds					32 bit Integer	Register Long
Other 68008	8 Mhz	OS-9	68K	...	18.0	...9.0
MUSTANG-08	10 Mhz	OS-9	68K	...	9.8	...6.3

C Benchmark Loop

```
Main()
{
        /* int I; */
        register long I;
        for (I=0; I < 999999; ++I);
}
```

Now even faster! with 12 Mhz CPU

C Compile times: OS-9 68K Hard Disk	
MUSTANG-08 8 Mhz CPU	0 min - 32 sec
Other popular 68008 system	1 min - 05 sec
MUSTANG-020	0 min - 21 sec

25 Megabyte Hard Disk System

$1,998.90

Complete with PROFESSIONAL OS-9 includes the $500.00 C compiler, PC style cabinet, heavy duty power supply, 5" DDDS 80 track floppy - Ready to Run

Unlike other 68008 systems there are several significant differences. The MUSTANG-08 is a full 12 Megahertz system. The RAM uses NO wait states, this means full bore MUSTANG type performance.

Also, allowing for addressable ROM/PROM the RAM is the maximum allowed for a 68008. The 68008 can only address a total of 1 Megabytes of RAM. The design allows all the RAM space (for all practical purposes) to be utilized. What is not available to the user is required and reserved for the system.

A RAM disk of 480K can be ea ily configured, leaving 288K free for program/system RAM space. The RAM DISK can be configured to any size your application requires (ystem must have 128K in addition to its other requirement). Leaving the remainder of the original 768K for program use. Sufficient source included (drivers, etc.)

FLEX is a trademark of TSC MUSTANG-08 is a trademark of CPI

Data-Comp Division

"A Decade of Quality Service"

Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

* Those with SWTPC Hi-density FLEX 5" - Call for special Info.

By: Dr. E. M. 'Bud' Pass
1454 Latta Lane N.W.
Conyers, GA 30207
404 483-1717/4570
Computer Systems Consultants

INTRODUCTION

This chapter continues the discussion of the conversion of Technical Systems Consultants BASIC and Microware BASIC09 programs into C programs begun in an earlier chapter.

CONVERTING BASIC PROGRAMS TO C

The OPEN statements perform essentially the same functions in TSC BASIC and in BASIC09; however, the formats for the statements are different. All require a file number, a file specifier, and a mode, which are all handled differently in the two types of BASIC interpreters.

TSC BASIC supports the following formats for OPEN statements:

```
 OPEN OLD filespec AS filenumb
OPEN NEW filespec AS filenumb
OPEN     filespec AS filenumb
```

Mode OLD requires the disk file to pre-exist and opens it for input. Mode NEW always creates a new file, deleting any old one by the same name, and opens it for output only. The null mode opens an existing file or creates a new one, and opens it for random access, both input and output.

BASIC09 supports the following formats for OPEN and CREATE statements:

```
OPEN     #filenumb, filespec[:options]
CREATE   #filenumb, filespec[:options]
```

OPEN opens an existing file and CREATE opens a new file. The options are as follows:

```
READ     open file for input only
WRITE    open file for output only
UPDATE   open file for input and output
EXEC     open file with execute permissions
DIR      open directory file
```

and they may be combined by separating the options with "+" symbols. All BASIC09 disk files are accessible in a random mode.

BASIC09 requires that file numbers always be preceded by a "#" symbol, whereas TSC BASIC requires that they be preceded by a "#" symbol in the INPUT, PRINT, GET, and PUT statements, but not in the OPEN and CLOSE statements.

Unless the low-level input/output processing represented by the C functions open, close, read, write, etc. is appropriate, the high-level input/output processing represented by the C functions fopen, fclose, fread, fwrite, etc. must be used. For these functions, file pointers, rather than file descriptors, must be used.

Although the INPUT statements are similar in format in TSC BASIC and in BASIC09, there are differences. The TSC BASIC INPUT statement assumes a comma separator between fields. The BASIC09 READ statement assumes a hex-00 separator. Both use carriage returns as record delimiters. Corresponding to the BASIC09 READ statement, is the BASIC09 WRITE statement, which separates fields with hex-00, and may be used as an alternative to the PRINT statement, which separates fields with spaces. The TSC BASIC INPUT statement always starts with a new record. The BASIC09 READ statement starts with the next byte after the previous one processed. The TSC BASIC INPUT LINE statement allows direct input without delimiters. The INPUT statements of both languages accept commas as delimiters. The BASIC09 GET statement, which may sometimes be used in a similar manner to the TSC BASIC INPUT LINE statement, does not honor backspaces and other control characters.

There is no direct equivalent in C to the TSC BASIC INPUT or INPUT LINE statements or the BASIC09 READ and GET statements. on an individual basis, these statements may usually be recoded as fscanf functions with appropriate control string, although the equivalence is not exact.

There are several minor differences between the TSC BASIC and BASIC09 PRINT statements. One has already been discussed, concerning the TSC BASIC use of file number zero corresponding (somewhat) to the BASIC09 use of file number one. Another is the lack of a parameter on the BASIC09 POS function, indicating that it may only be used in a PRINT statement, and only to designate the character position in the current output buffer.

A major difference between the TSC BASIC and BASIC09 PRINT statements concerns the interpretation of the control string in the PRINT USING option. The TSC BASIC PRINT USING control string may contain control sequences similar to the following:

!	single character string field
\...\	multiple character string field
####.##	number field
$$####.##	number field with floating dollars
**###.##	number field with floating asterisks
#,,,##.##	number field with inserted commas
####.##-	number field with trailing minus
#.#####^^^^	number field with scientific notation
(other)	inserted character

in which repeated symbols designate field width and number of decimal places. The BASIC09 PRINT USING control string may may contain control sequences similar to the following:

Bwj	boolean format
Ew.fj	exponential format
Hwj	hexadecimal format
Iwj	integer format
Rw.fj	real format
Swj	string format
Tn	tab to column
Xn	insert spaces
'string'	inserted character string

where "w" represents total field width, "f" represents fractional field width, "j" represents justification mode ("<" is left justification, ">" is right justification, "^" is right justification with trailing sign), and "n" represents an integer value.

Obviously, there are differences and similarities in the PRINT USING interpretation which must be considered when converting BASIC PRINT USING control strings to C fprintf control strings. Since TSC BASIC and BASIC09 both support dynamic or static construction of the control strings, the changes required may be simple or complex, and will not in general be exact, since TSC BASIC PRINT USING control strings may specify operations (such as dollar sign or asterisk insertion) not supported in C printf control strings. However, in most cases, the changes required to convert the control strings will be straightforward, and it will not be necessary to code an interpreter to handle the PRINT USING control strings directly.

The TSC BASIC function INCH$(filenumb) generally corresponds to a BASIC09 GET#filenumb,char$ sequence. One conversion problem with these statements concerns the fact that the characteristics of the OS9 device driver may affect the value returned and whether or not a character input from the keyboard is echoed to the screen.

TSC BASIC supports record-oriented random disk I/O, whereas BASIC09 supports byte-oriented random disk I/O. The logical means of converting from the TSC BASIC style of random I/O to the C style of structured I/O is to convert the BASIC records to C data structures. Unfortunately, there is no simple method to perform this conversion which will work in all cases. The TSC BASIC FIELD statements define the record formats in a dynamic manner, so that the description of a record may vary as the program proceeds. BASIC09 supports a complex data structure, but it is not dynamic.

Given that a C complex data structure can be established to describe a record, the C I/O statements can be used to perform the record-oriented I/O. The problem is how to convert FIELD or record statements and operations on FIELDed or record variables into C structures and operations on structure elements.

One manner in which to accomplish this for TSC BASIC is to define one C complex data structure for each file as one string of length 252. Then each original TSC BASIC FIELDed variable would be represented by two C variables, each set at the location of the original FIELD statement, providing starting position and length of the original FIELDed variables in a dynamic manner. A naming convention for these variables would relate them to the original FIELDed variables. References to the value of FIELDed variables would be made with the C equivalent of the BASIC MID$ function. However, C does not directly support BASIC string operations, so this approach could become complicated.

Another manner in which to define the structures, which is applicable only in the case of fixed FIELDs, is as composed of the original FIELDed variables, but defined as strings with location and length the same as in the original record description. Then the FIELDed variable assignment statements LSET and RSET are much simpler to convert to C.

The TSC BASIC string conversion functions CVTxx do not exist in C and their use must be avoided by modification of structure definitions for FIELDed variables or defined as C functions.

The TSC BASIC concept of virtual arrays is very powerful and is used by many programs. Unfortunately, C has no direct analog for virtual arrays, considerably complicating their conversion. In reality, virtual arrays are a notational convenience for fixed length, record oriented random disk access. The DIM#filenumb statements establishing them may be changed to DIM and FIELD statements, and each occurrence of their use may be replaced with BASIC code to locate and read or locate and write the indicated elements.

In the case of reasonably small virtual arrays, they may be changed to string arrays and handled by reading the entire file into memory during initialization and then writing it from memory during wrapup. Whichever approach is chosen, the modifications should be made and checked out before conversion, not afterward.

Error Handling

TSC BASIC and BASIC09 provide similar structures for the interception and handling of many types of errors. Error interception routines are established with the "ON ERROR GOTO label" statement, which specifies that control is to be transferred to label "label" if an error occurs. TSC BASIC error interception is cancelled with the "ON ERROR GOTO [0]" statement, and BASIC09 error interception is cancelled with the "ON ERROR" statement (without the "GOTO label").

Both TSC BASIC and BASIC09 place the error number into pseudo variable ERR: however, BASIC09 zeroes ERR after the first reference, but TSC BASIC does not zero ERR automatically. TSC BASIC places the line number on which the error was detected into pseudo variable ERL. BASIC09 has no such facility.

TSC BASIC requires that the exit from the error handling routine be accomplished with the RESUME statement, which can designate a specific label to which to return (with the "RES ME label" option), or can return to the statement at which the original error occurred (with the "RESUME" option). BASIC09 does not require any particular means of exiting from an error handling routine, but provides no direct means of returning to the line on which the statement at which the original error occurred.

Unfortunately, C's error processing is very different from BASIC's. The error number and error line facilities of BASIC have little direct analog in C. If it is supported, the signal function of many of the standard C libraries is used to establish interrupt handlers for specified error conditions and the errno variable is used to specify the error number. Thus, the conversion of error handling facilities may be quite involved.

Functions

Function calls provide much of the notational power of the BASIC language, and both TSC BASIC and BASIC09 support function calls, although in somewhat different means and with different restrictions.

TSC BASIC supports the DEF statement, which allows the definition of a short (one-line) function with parameters, similar in meaning to the C #define macro.

For an example of the problems of expanding expressions to include DEF functions, consider the following TSC BASIC program fragment:

```
DEF FNA(X)=LOG(-X)+1
:
:
Y=2*FNA(Z-1)
```

The direct substitution of the body and parameter of function FNA would produce the following incorrect results:

```
Y=2*LOG(-Z-1)+1
```

whereas the following statement represents the correct conversion of the original statement:

```
Y=2*(LOG(-(Z-1))+1)
```

so #define parameters and bodies should be enclosed in parentheses (when necessary) to maintain the integrity of the original expressions.

C requires that the correct type of function be used. Usually, this implies that the type of function agree with the type of argument.

This discussion is continued in the next chapter.

EXAMPLE C PROGRAM

Following is this month's example C program: it outputs a UNIX shell script consisting of chown, chgrp, and chmod commands which will correctly establish groups, modes, and ownership of all files and directories included in and below the directory-list. It could be readily modified to do the same for OS-9 or UNIFLEX.

```
#include <stdio.h>
#include "version.h"

FILE *dir;
char *p, path[256], string[256], temp[64], work[16], *strrchr();
short int bl, i;

main(argc, argv)
int argc;
char **argv;
{
#ifndef IBMPC
        if (argc < 2)
        {
                fprintf(stderr, "Usage: %s directory-list\n", argv[0]);
                fprintf(stderr,
                        " %s outputs a shell script consisting of\n",
                        argv[0]);
                fprintf(stderr, " chown, chgrp, and chmod commands\n");
                fprintf(stderr, " which will correctly establish,\n");
                fprintf(stderr, " groups, modes, and ownership of all\n");
                fprintf(stderr, " files and directories included in\n");
                fprintf(stderr, " and below the directory-list.\n\n");
                exit(0);
        }
        for (i = 1; i < argc; ++i)
        {
#ifdef SYSTEM5
                sprintf(string, "ls -ld %s >%s", argv[i], tmpnam(temp));
#else
                sprintf(string, "ls -ldg %s >%s", argv[i], temp);
#endif
                if (system(string))
                {
                        fprintf(stderr, "Cannot find directory %s\n",
                                argv[i]);
                        break;
                }
                if ((dir = fopen(temp, "r")) && fgets(string, 256, dir))
                {
                        genscript(string):
                        fclose(dir);
                        unlink(temp);
#ifdef SYSTEM5
                        sprintf(string, "ls -lR %s >%s", argv[i], temp);
#else
                        sprintf(string, "ls -lgR %s >%s", argv[i], temp);
#endif
                        system(string);
                        sprintf(path, "%s/", argv[i]);
                        if (dir = fopen(temp, "r"))
                        {
```

```
                                while (fgets(string, 256, dir))
                                        genscript(string);
                                fclose(dir);
                                unlink(temp);
                        }
                        else
                        {
                                fprintf(stderr,
                                        "Cannot open temp file %s\n",
                                        temp);
                                break;
                        }
                        unlink(temp);
                }
                else
                {
                        fprintf(stderr, "Cannot read temp file %s\n",
                                temp);
                        break;
                }
        }
#endif
        unlink(temp);
        exit(0);
}

genscript(string)
char *string;
{
        *(p = string + strlen(string) - 1) = 0;
        if (!*string)
        {
                ++bl;
                return;
        }
        if (bl && (*--p == ':'))
        {
                bl = *p = 0;
                sprintf(path, "%s/", string);
                return;
        }
        bl = 0;
        switch (*string)
        {
        case '-':
        case 'b':
        case 'c':
        case 'd':
        case 'l':
        case 'p':
                if (p = strrchr(string, ' '))
                {
                        work[8] = 0;
                        printf("chown %s %s%s\n",
                                strncpy(work, string + 15, 8), path, ++p);
                        printf("chgrp %s %s%s\n",
                                strncpy(work, string + 24, 8), path, p);
                        printf("chmod u=");
                        if (string[1] == 'r')
                                printf("r");
                        if (string[2] == 'w')
                                printf("w");
                        if ((string[3] == 'x') || (string[3] == 's'))
                                printf("x");
                        if (string[3] == 's')
                                printf("s");
                        if ((string[9] & 0x5f) == 'T')
                                printf("t");
                        printf(",g=");
                        if (string[4] == 'r')
                                printf("r");
                        if (string[5] == 'w')
                                printf("w");
                        if ((string[6] == 'x') || (string[6] == 's'))
                                printf("x");
                        if (string[6] == 's')
                                printf("s");
                        printf(",o=");
                        if (string[7] == 'r')
                                printf("r");
                        if (string[8] == 'w')
                                printf("w");
                        if ((string[9] == 'x') || (string[9] == 't'))
                                printf("x");
                        printf(" %s%s\n", path, p);
                }
        }
}

EOF
```

FOR THOSE WHO NEED TO KNOW

68 MICRO JOURNAL™

Basically OS-9

Dedicated to the serious OS-9 user.
The fastest growing users group world-wide!

6809 - 68030

A Tutorial Series

By: Ron Voigts
2024 Baldwin Court
Glendale Heights, IL

ALTERNATIVES TO THE EDITOR, THE WORD PROCESSOR

Some months back I did a report on using editors. Specifically, I used the OS-9 editor. It contains many of the features found in editors. There are other editors out there, which will act similarly. In many cases you will probably find one that is more to you liking. Sometimes, the editor is part of another piece of software. An example of this would be BASIC09. It has a number of different modes, one of which is its own editor.

There are however other ways to get you text into a file. Namely there is the word processor. Some may feel that the word processor is another editor. I like to think of them as separate, but part of the same family. If you look at the evolution of software, they are both part of the same branch, but have split into two separate families. (In many cases, a blend of the two may exist. But we will consider them separate entities.)

One major difference between the two is how data is entered. The editor is line oriented. Information is entered on a line-by-line basis. Some editors can split and merge lines easily, but basically entering a line is the how it works. In fact, some of the early editors were line number oriented. I originally started on one like that. Later when it came time to switch to one without line numbers, I thought I would never be able to make the transition. Now I don't think I could use one with numbers again.

The word processor is screen oriented. Basically, you move around the screen and put text where the cursor lands. If you move too close to the right hand margin, it wraps around to the next line carrying the last word with it. Some specialty keys have been added to move the cursor around or move the screen. As text is typed, the cursor moves along with everything in front of it moving along also. There may be an overstrike mode. Here the entered text writes over the old. But the textq entry is always oriented from the screen.

Because of the screen orientation, there is a great reliance on the terminal's screen control characters. The usual ones are backspace, home cursor, clear screen, and clear to end-of-line. Some terminals allow moving the cursor using x and y coordinates. The entire idea is to give the illusion of mobility on the screen. Usually when a word processor is acquired, it must be configured for the particular terminal it is used on. In contrast, the editor moves unidirectionally and usually works on any terminal without any previous adjustments.

Word processors provide specialty keys. This may be on a key pad. They could be the standard keys, but used differently. For example, holding down the CTRL key may cause the keys to have new functions. Many of the new systems are ICON operated. (More about it later.) No matter how it is done the net result is the same.

Rather then talk about hypothetical word processors, it is best to look at a real one. I use mostly STYLOGRAPH by Stylo Computer. They make the word processor for both OS-9 Level I and II, as well as, OS-9 on the 68K system. An interesting feature of Stylo is that it dynamically updates the screen as it is used. The screen appears as it will be printed. This includes all the pagination. Many word processors add this as a final step before printing. Many times they are a separate program. For our discussion we will focus mainly the text entry capabilities.

There four modes to Stylo. They are insert mode, supervisor mode, overwrite mode, and editor mode. The supervisor mode is the main menu, providing for things like loading files, saving files and printing to the printer. Overwrite mode allows typing over text that was previously entered. This can be useful. However, we'll consider the two remaining modes in more detail.

For entering text, there is the insert mode. This one adds text to the current location. Text after the cursor location is moved ahead. In this mode the CTRL key must be used to access any special features. The following are available when used with the CTRL key.

W Delete word
D Delete letter
R Set Tab
T Tab
U Underline
O Overline
P Page Status
A Assistance
F Format Display
G Ghost Hyphen
K Subscript
X Delete Line
V View Mods
B Bold face
N Name Error
Y Superscript

I won't go into great details on these, except to highlight a few of them. Notice that lines, words and characters can be easily deleted. There are also many text formatting features included like underline and subscript. Name error reports the last error. Assistance displays help screens of information.

requires an input at some command line. With the word processor, it is a simple key stroke. The editor has little in the way of text formatting. Usually help comes from reading the manual and not online, as it is here.

Stylo has what it calls the "Editor Mode". This mode permits text, screen and cursor manipulations. Among its features are:

1 Go to Overwrite mode
7 Scroll screen left
9 Scroll screen right
S Save a part of text
W Withdraw the text from the buffer
D Duplicate the buffer
F Find a target string
Y Move cursor a word left
U Move up a line
O Move up a screen
P Move to a new page
[MOve a word right
J Move cursor left
K Move cursor to either margin
L Move cursor right
M Move down a line
. Move down a screen

These give an idea of what can be done while in the editor mode. Notice that most of them are screen oriented maneuvers. The feeling is that you're moving around a page, rather then from line-to-line.

This has been a whirlwind tour of word processors and Stylo. Stylo comes with a very thick manual and has much more than I have told here. There are many things about it and word processors, in general, that could not be covered in the space of one column.

Another thing, I have drawn a sharp distinction between the editor and word processor. The world of text processors is not black and white. There are many that come somewhere between the two extremes. Many text processors consider themselves to hybrids, combining the best of both worlds. If you use have used editors and word processors, you have probably made your mind up which you prefer. I find most people prefer a screen oriented environment. There are many fine word processors and screen oriented editors available.

THE FUTURE OF THE WP AND EDITOR

As I pointed out earlier, ICON driven test processors are available. Many of the computer systems that have graphic capabilities use ICONs. The usual way of doing things is to move a mouse, joy stick or track ball, positioning an arrow over a picture that describes what is to be accomplished. If more information is needed, another menu appears, requiring further response. The idea is to provide an environment that requires little knowledge. The system takes you by the hand and leads you to where you want to go.

I know one person that has a system that is capable of accepting command lines or ICONs. He prefers using command lines. Another friend uses the ICONs and wouldn't have any other way. He says, he doesn't have time to learn things from a long manual. Personally, I like to use commands lines. However, using the ICONs is a refreshing change and can be fun.

One item that I have not seen in OS-9 is a language oriented editor. This is something like the built in editor of BASIC09. The difference is that the editor can be biased toward a particular language. I have one at work that I use. It interprets the file name and orients itself accordingly. If the program name ends in .C, it knows I am writing a C program. If it ends in .BAS, I am writing BASIC. It also understands FORTRAN and PASCAL.

The editor generates programming templets. These are outlines of how to program in the particular language. Let us say your are using this editor. It prints a line like:

```
<statement>;
```

The cursor is positioned over the above. The Expand Templet key is pressed and a menu appears giving possible choices. So, we select the WHILE construction from it. The above line expands to:

```
while ( <expression> )
   <statement>;
```

Now we further continue this growth process and expand the Templet more. Eventually, customized inputs must be used. But this type of activity can help someone get started writing programs.

Another advantage of the language oriented editor is that it contains help for using the language. Again in our previous example, what if we did not know what WHILE did. With this type of editor, we would move the cursor over the word in question and type the Language Help key. Instantly a description of how to use the particular word would appear.

Another item that is handy to have while editing text is an online dictionary. I have seen one word processor that had this feature. Unfortunately, it was not written for OS-9. To use this, one moved the cursor over the word, spelled it as close as is possible. A Look Up key was pressed and the word processor compared the word to words in the dictionary. Close matches were displayed. The user could then select the correct spelling. (Hopefully, it was found!)

These are some thoughts of my mine about what I would like to see in a word processor or editor. There is so much that can be done. I have only touched on a few ideas. I am sure there many more out there. It only takes an idea and someone willing to make it work.

GETTING OPTIONS

I got a letter from a reader recently who ran into an interesting problem. He had copied a program from the column and found that it crashed his system. The line he reports that did it is:

```
while ( --argc>0 && (*--argv)[0]=='-' )
  <statement> ;
```

The problem, he reports, is the second expression is checked, even if the argc equals 0. Now, Kernighan and Ritchie in The C Programming Language tell, this line should be evaluated left-to-right. If an expression is FALSE, no more evaluations occur and the following statement is not executed. We could argue the point. Instead a better method to parse the input line could be found.

I present here my version of getopt(). It has similarities to the one described by Dr. 'Bud' Pass in his may column. I

To accomplish many of these operations with an editor also took a look at ones that are available with some of the newer C compilers. And I added my own ideas. If you like this one add it to your C library. Add improvements if you like. As time goes on, I will probably use it my C programs.

I plan on explaining how to use getopt(). I will let the workings of it to you. I think once you understand what it does, how it works will be clear. Here's the classical manual type description.

```
extern char *optarg; /* Option argument */
extern int optn; /* Next option */
extern opterr; /* Error status */

char *getopt( c, v, optlist )

int c;  /* argument count */
char **v; /* argument vector */
char *optlist; /* option list */
```

Getopt() returns a pointer to the next option. When NULL is returned, no more options are available. An option must be preceded by '-'. Encountering anything else signals the end of options. The pointer returned is the option in the parameter list. There may be other characters following it. C is usually the argument count. V is the argument vector.

On entry optn should point to the first option, usually 1. After each call optn will be incremented, pointing to the next option.

Optlist is the option list. It contains the expected option characters. If an option character is followed by a '=', an argument is sought. The option's argument can immediately follow the option or be separated by a space. An equal sign can also be used for improved readability.

Successful calls to getopt() will return with opterr equal to 0. Should an error occur, a negative number will be returned. A -1 is Illegal Option. This is one not found in optlist. A -2 is Missing Option Argument. This occurs when an option argument is expected and the end of the input line is encountered. A -3 is Argument Not Expected. This happens when it appears that an option argument has been passed, but none was expected.

So now you are ready to use getopt(). Right? Well let me show you how to use it. Listing 2 is little program that will help demonstrate getopt()'s uses. First, create the listing shown in Listing 1. Make certain that it is in the working directory. Next, create the program in Listing 2. Call it optest.c. Now, compile it! The line:

```
#include "getopt.c"
```

will cause the compiler to load it. This is a simple way to include outside program parts.

We are now ready to try it. Notice that a, b, c and d have been defined as options. The = following c and d mean that they expect some type of argument. So we are ready to go. Try this line.

```
optest -a -c hello -d=bye -f -c
```

You should see the following on the screen.

```
opt=a optarg= opterr=0 optn=2
opt=c optarg=hello opterr=0 optn=4
opt=d optarg=bye opterr=0 optn=5
opt=f optarg= opterr=-1 optn=6
opt=c optarg= opterr=-2 optn=7
```

Here a is a legitimate option and returned first. Next is c. It is not followed directly by an argument, but one is expected. So "hello" is picked up. D comes along with its argument "bye". F is not a legal option, so its returned with a error, -1. C comes last, but this time there are no arguments where one was expected. So an error, -2, is returned. Optn is pointing to the next line argument. This can easily be accessed by its pointer, argv[optn].

Eventually, you will want to make it part of your personal library of C routines. You will want to create a object file, using:

```
cc1 -r getopt.c
```

This will create an object file, called getopt.r. Next you will want t put it into your library.

```
copy getopt.r /d1/LIBRARY/getopt
```

And now we are ready to use. One thing remains. Some variables must be declared as external. These are done for you in Listing 3. You'll want to put them in another file. Call it getopt.h. When you write programs using getopt() add to the beginning of your source code:

```
#include "getopt.h"
```

When you are ready to compile use a line like:

```
cc1 file.c -l=/d0/LIBRARY/getopt
```

This will instruct the C compiler to link to the object module in /d0/LIBRARY (providing it is on D0).

Try out getopt() and let me know what you think. Make improvements, if you want. If you come up with something worth sharing, drop me a line. We'll put it in the column.

Until next time have fun!

LISTING ONE

```
/* *************************

   Name: GETOPT
   By: Ron Voigts
   Date: 25-MAY-87

   *************************

   Function:
   This function examines the argument list
   returning a pointer to the option and
   its argument.  A null string is pointed
   to if the option has now argument.

   *************************

   Version 1.00
   Original.

   ************************* */

#define TRUE 1
#define FALSE 0

char *optarg; /* Option argument */
int optn;     /* Next option     */
int opterr;   /* Error status    */

```

```
char *getopt( c, v, optlist )
int c;          /* argument count  */
char **v;       /* argument vector */
char *optlist; /* option list      */

{
   int isoption;    /* option flag  */
   int hasarg;      /* option argument flag */
   register int i;  /* useful index */
   char *opt;       /* option pointer */
   char *t;         /* argument pointer */
   static char *null='\0'; /* null string */

/* Set up the null string for 'optarg' */
   optarg = null;

/* Set up the error return status */
   opterr = 0; /* No errors */

/* Set up the argument */
   t=v[optn];

/* We are at the end of the argument list */
   if ( (optn==c) || (*t!='-') || ( *t=='-' && *(t+1)=='\0' ) )
      return( 0 );

/* We can set the option */
   opt = t+1;

/* Check if we have an option with an argument */
   isoption = FALSE;
   hasarg = FALSE;
   for ( i=0; i<strlen(optlist); i++ )
      if ( toupper(*(t+1))==toupper(optlist[i]) ) {
         isoption = TRUE;
         if ( optlist[i+1]=='=')
            hasarg = TRUE;
      }

/* If this is not an option then return with error */
   if ( !isoption )
      opterr=-1; /* illegal option */

/* Now we check and set up the argument */
   if ( hasarg ) {
      if (*(t+2) == '\0')
         if ( optn < c-1 )
            optarg = v[++optn];
         else
            opterr=-2; /* Missing option argument */
      else
         optarg = t+2;
      if ( *optarg=='=' )
         optarg++;
   } else
      if ( *(t+2) != '\0' )
         opterr=-3; /* Argument not expected */

/* Now we have an argument and option */
   optn++; /* Adjust the next pointer */
   return( opt ); /* Return the option pointer */

}

```

LISTING TWO

```
/* **************************

   Name: OPTEST
   By: Ron Voigts
   Date: 7-JUN-87

   **************************

   Function:
   This little routine can be used
   to test the function getopt().

   **************************

   Version 1.00
   Original.

   ************************** */


#include "getopt.c"

main( argc, argv )
int argc;
char **argv;
{
    char *opt;
    char *list="abc=d=";

/* Set this to 1 */
    optn=1;

/* Now we'll scan the input line */
    while ((opt=getopt(argc, argv, list )) != 0 ) {
       printf("opt=%c ", *opt );
       printf("optarg=%s ", optarg );
       printf("opterr=%d ", opterr );
       printf("optn=%d \n",optn );
    }

}

```

FOR THOSE WHO NEED TO KNOW

68 MICRO JOURNAL™

SOFTWARE USER NOTES

A Tutorial Series

By: Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

From Basic Assembler to HLL's

Son of Mustang

A few weeks ago the company's single board computer, the Peripheral Technology board with the 68008 arrived, followed shortly by OS-9 and SK*DOS. We fired the system up with a couple of 80 track drives and were able to run both operating systems immediately. A few days later I decided to get a hard disk attached to the system since we had a spare and a WD 1002-05 controller that we had bought used recently. The 20 Mbyte unit was not exactly the same as the configuration (device descriptor) file in the OS-9, so we set off to try it out first with SK*DOS, just to try to keep things simple. First problem was that PT uses the second drive select from the WD, as drive 0 and the third as drive 1. Once we understood that (thanks to a call to Peter Stark), we got the hard disk formatted and copied all of the SK*DOS files over to it. The SK*DOS HDFORMAT utility prompts for the number of cylinders, heads, sectors per track, etc., so it will handle any hard disk drive provided you know what the parameters are. All the hard disk drive manuals we have, contain all the necessary information.

Our next step was to try to format about 1/3 of the disk under SK*DOS and the remainder for OS-9. After a day's worth of fooling around with files linked to files linked to other files in OS-9 and some experimenting because the meaning of the parameters was abundantly unclear, we managed to get the remainder of the disk formatted under OS-9, and then to get the boot properly installed so we can now boot either SK*DOS or OS-9 on the system. I installed PAT/68000 (OS-9 Version) and we were immediately able to run it and edit text files. We loaded my preliminary copy of PLuS from Windrush and it ran first try.

I am working on a PLuS version of PAT since I decided it would be easier to link to SK*DOS than the "C" version. I hope to get PAT running under SK*DOS, though it will probably be a few months before that spare time effort is completed and debugged. Meanwhile Pete sent me an assembler (thanks to Bud Pass), a TINY "C" compiler, (miniscule would be a better word), and a screen editor written in "C" to use to get other things moved into SK*DOS. I thought I would try to get JUST running in the tiny "C", but I soon found that it didn't allow pointers, didn't support the logical operators ! (not), &&, and ||. Of course it didn't understand what a static variable declaration meant, and didn't support initializers either. Then I found that FOR NEXT loops are not implemented. I'd have to rewrite half of the program. I decided to wait for the nearly full "C" compiler that is in the mill.

Having struck out on doing much more with software for the moment, I decided that we would take this rather nicely running system and put it in a box. We had ordered an IBM Clone box with an IBM clone power supply in it, and about that time the box arrived, so we put the processor and hard disk controller in a stack with long spacers between, and then installed the power supply, an 80 track floppy drive and the 20 Mbyte hard disk drive. The system would run seemingly for an hour or so and then it would bomb. We figured we had a heat problem and added a small fan blowing directly on the processor and disk interface card. Things got better, but still every hour or so the system would bomb with a ridiculous error message, indicative of the memory having lost or modified data. We separated the processor and disk interface boards further with longer spacers with little change in the problem. I added a shield (aluminum foil in an envelope, grounded with a clip lead) between the boards and thought things were a little better, but the problem still occurred. We substituted the power supply that we had been using when the system was scattered all over the bench, and still no change. Finally in desperation, I unfastened the processor board, which was on top of the controller, and moved it a few inches away. The problem went away, at least for the remainder of the day, and since it had not gone that long previously without failure, I assume that the problem was in the proximity of the two boards. We will mount the processor well away from the disk controller, and things should be fine. **See Editor's Note, end of this article.

Hard Disk Woes

We have been using hard disk drives for about three years now, and we had a 20 Mbyte drive that ran just fine for a year or so and then suddenly quit writing. It would read fine, boot, etc. but any attempt to do an operation that required writing to it resulted

in instant error messages. The problem would leave as quickly as it had come, and it has been an intermittent problem for some time. Not too long ago it quit and seemed determined to stay in that mode. We explored having it repaired, but I was discouraged by prospects of paying about the cost of a new unit for a repair. I decided one day that the problem had to be an intermittent connection since when it worked it worked fine. The fact that it "fixed itself" ruled out the failure of an integrated circuit (at least a burn out failure). I decided that I would work on what I could work on without opening the sealed portion of the drive, so I carefully went over the PC board, soldering all the connections. I didn't find any obvious cold solder joints, but when we hooked the drive up to the new 68008 computer, it worked instantly and has been running for days with no problems. Perhaps I have fixed it.

How to Wipe Out a Hard Disk Drive Instantly

This is being written about a week after the above. That 20 Mbyte disk is still working perfectly. I think the problem has been cured permanently. However...as the old saying goes, "you win some, you lose some". We had three hard disk drives, all used, that had been sitting around our work area for some time, that we had never gotten around to connecting to any computer. Two are old Segate 506 units and one of them has a "dead" tag attached to it. Given the ease of telling SK*DOS the disk parameters, I thought I would check them out. The dead one really was. It didn't do anything...Maybe it is repairable with a little fiddling. The second seemed to work. It formatted perfectly, but when I tried copying files to it, it indicated a bad sector on track 0, and once the directory got to that sector I got a 'DRIVE NOT READY' error when I tried to access it. I will see later if I can figure out how to trick it into working.

So far, nothing lost...I said I'd pay for the drives if they were usable. Then I dug out a newer 10 Mbyte drive and proceeded to connect the power with the +5 and +12 volt supplies reversed. I figured I had literally blown it as soon as I discovered my error. I made the correction and tried it out anyway. It accelerates to speed and goes through its self test mode, so things couldn't be extremely bad, but it doesn't talk to the interface. At least I am a little ahead, since I have repaired a 20 megabyte drive and destroyed a 10. Later the same day, I turned off a development system after carefully removing the disks from the 8" floppy drives, but I forgot a disk in one of the 5" drives and managed to clobber a sector right in the middle of an important program source file. Fortunately there was a recent backup, and later I managed to recover the file and fix the two corrupted characters, but I certainly didn't feel very good about my day.

More OS-9

A few months ago I expressed some displeasure with the OS-9 manual, particularly as a beginning user of OS-9. I mentioned that the manual begins very early to talk about RBF devices and SCF devices, but never defines either. Well, I don't think I lied, but deep down in the OS-9 Technical Manual are two sections describing files, titled "Random Block Files" and "Sequential Character Files". I assume that the abbreviations RBF and SCF are for those, though they are not used in the sections with those titles.

The other day as I was starting to translate PAT FLEX 6809 PL/9 version to PAT PLuS 68000 OS-9 version, I decided to get into the file handling early on, since it would be the major difference between the FLEX and the OS-9 version. I found that between the OS-9 Programmer's Manual and the PLuS manual, I was able to figure out how to get filenames from the command line in only a couple of hours and a dozen tries at a test program. It only took another 8 or 9 tries to figure out how to detect the End of File error message and use it to terminate reading of an open text file. Golly, only 3/4 of a day and I can now write a PLuS program to copy a file under OS-9! I'll keep you all posted on progress in the 68XXX area. It is beginning to feel like it did in the early days of the 6800 and 6809.

Speaking of OS-9 and its documentation, I must describe my adventure game trying to learn something about it. Microware "C" is a pretty standard "C" compiler, and as such, it pretty well hides the operating system from the user with regard to file handling. PLuS, however, like PL9 uses the operating system rather directly, and simply assists the user in accessing the OS. The other day, I started working on the PAT translation to PLuS, and was busily digging into the file handling to see which procedures in files.lib would match those in the FLEX.LIB of PL9. I thought I was stuck for a while when I came upon the necessity for a "procedure rename" to allow me to rename a file (add a .bak extension or change the existing one to that). The PLuS os9.lib has a procedure called f_fork, and I saw that it was designed to do the OS-9 F$Fork system call, supplying at least some of the parameters automatically and stuffing the user supplied ones into the appropriate 68000 registers. *Unfortunately, the folks at Windrush fell into the trap of being only as descriptive as the OS-9 technical manual.* Let me quote from their manual.

"asmproc
f_fork(long,long,long,long,integer,integer);

Create a new process using the OS-9 F$Fork system call. The
parameters are (in order):

LONG A pointer to the module name
LONG A pointer to any parameters for the module
LONG The parameter size
LONG The additional memory required
INTEGER The no. of I/O paths to copy
INTEGER The process priority

Well, maybe I'm dumb but about the only things more or less clear to me in the above are the first two items. I decided that perhaps the first long, had to be a pointer to a text string containing the module name, and I wanted to call the rename function. Since including a string as a parameter in PLuS passes a pointer to that string, and it is terminated with a null automatically, I was on my way.

f_fork("rename",

The second parameter, I reasoned further probably was a pointer to a string containing the necessary parameters for the rename call, in this case an old filename and a new filename separated by

a space. Since I was writing a test program, I wrote code to get the filenames from the command line (see the other adventure story above) and assemble them into a string with a space between them in an array of characters called buffer. In PLuS, a "dot" before a variable name makes it a pointer to that variable.

f_fork("rename",.buffer

The next item to be passed is "the parameter size". How big is a parameter? If I had been correct about pointers to strings, this probably means the string length. I "included" strsubs.lib, and used strcat to put the filenames together, and strlen to return a value for the string length, and put it in a variable called length. (Later I found I had to add 1 to the value obtained by strlen or my rename was short the last character of the new filename).

f_fork("rename,.buffer,length

The last LONG parameter is "The additional memory required". I looked at that and asked myself "for what?" Are we talking about the module size? A dir -x -e rename got me a module size of 3236 bytes. Maybe they are also talking about stack space for the module's execution. How would I know that. Must be the module code byte count.

f_fork("rename",.buffer,length,3300

Now here's a nice description of what is wanted for the next parameter. "The no. of I/O paths to copy". I suppose if I were calling a program to print a file to a printer, and I had the printer path open in my program, I would use 4 (stdin, stdout, stderr, and printer). In the present case I was only using the terminal, and I would expect error messages on the terminal. I decided to try 3 and see what happened.

f_fork("rename",.buffer,length,3300,3

All that is left is the priority. Since it is an integer, it can't be greater than 32767, but maybe OS-9 has some much smaller priority limit. I looked up "priority" in the index and found a section on process priority. "Each process is given an initial priority that is specified in the password file. This priority is set by the system manager." The next paragraph continues "If you have a program that you want the system to give higher priority, the "^" modifier is used. By specifying a higher priority, a process will be placed higher in the execution queue. For example: $ format /d1 ^255".

Well, now I know that priorities may be as high as 255, but what are the maximum and minimum priorities? A check in the index of the technical manual found me a section that indicates that the lowest priority process that will be executed may be set by the system "super user" in a variable called D_MinPty, and goes on to say that that variable usually contains the value 0. Since I haven't fiddled with it, I can assume that any priority greater than zero will get my code executed sometime, but there is no mention whatever of a maximum priority. Since the variable type of the required parameter is an integer, I suppose 32767 would have to be an outer limit on the possible range of values. I decided to try 128. All that might have been academic anyway, since the next procedure discussed in the PLuS manual is f_wait. The text says "Wait for a child process to terminate, using the OS-9 F$Wait system call. I assumed that meant that the rename process would run completely before returning to my test program if I included that call, so I did so. Then I wrote a quick test program to rename a file, using my previously worked out get_filename procedure to get the names of the files from the command line. The program worked on the first try, except that the new name was missing its last character. I added 1 to strlen to obtain a value for length in the call, and all was fine.

```
f_fork("rename",.buffer,length,3300,3,128);
f_wait;
```

In all fairness, the OS-9 technical manual page describing the system call F$Fork does shed a little more light on the required parameters, indicating at least that the module name pointer is to point at a string. It does clarify the "additional memory" parameter as being memory over and above the module size and workspace. I found by further test that 0 works as well as 3300 for the additional memory value in my test program.

After a year of doing things on and off with OS-9, my observation simply is that the manual always tells the user what he can do, but almost always doesn't give complete information on how to do it. The writer of the OS-9 manual made too many assumptions about the user's familiarity with OS-9, probably because he was too familiar with it. After five years of talking about SCF's and RBF's, how could anyone not know what those abbreviations mean? Wake up, you folks at Microware! You've been talking about RBF's and SCF's and F$Forks and E$NEMod for so long these mnemonics are all part of your standard English vocabulary. To someone who hasn't seen OS-9 before, it's all Greek (or maybe I should say Vulcan)! In the case of trying to figure out the maximum priority that is allowed, not only did the primary reference to priority in the manual not yield the information, a look at ALL the references to priority didn't spell it out either. Your OS-9 manual might be a dandy reference guide for the folks at Microware who wrote it, but a user guide it is NOT.

I hasten to add that OS-9 is a fine operating system, and I will be using it for a long time to come. I expect I will learn a great deal more about it in the process. OS-9 has a large number of very nice features for an operating system with such a small kernel. The code size of OS-9 is miniscule compared to UNIX, for example. My only quarrel is with the supplied documentation.

My next project is to begin to learn 68000 assembler code. Now that I have SK*DOS so I don't first have to learn the intricacies of OS-9, maybe I have a chance. After getting a little experience with assembler code, I will be brave enough to tackle writing code to run under OS-9. As I make dumb mistakes, I'll pass along my experiences.

Update on PLuS

Last time, I did a little review on PLuS indicating that I had been debugging a preliminary version for Graham Trott and the folks at Windrush (I shouldn't say debugging, really, since what I was doing was only uncovering and reporting bugs). I now have version 3

in my hands. It crossed my last bug report in the mail. I'm sure, so it still has the last trivial bugs that I had reported last time. I expect that these will be repaired very soon, but even without their repair, PLuS is now a complete and pretty well debugged compiler. I have a large backlog of software in PL9 that now can be ported to the 68000 systems with little effort. Of course it is easier to write the simpler utilities from scratch again than to move them from another system. Don's treatment of last month's review has reminded me that many of you readers may be new to computing, and that I shouldn't make any assumptions that you all have been following this column for the past 5 years, so I really ought to do a full review of both PLuS and OmegaSoft Pascal starting right from ground zero.

A Fix for JUST

Last week my friend and contributor to '68' Micro Journal, Dr. Albert McDaniel phoned me to tell me about a great bargain he had just found on a couple of used terminals. While we were talking he mentioned that he had found out how to cause a peculiar bug in JUST to occur. We had noted for some time that occasionally for no known reason, JUST would decide that the line length should be 1, and it would suddenly put one character on a line when outputting text. Albert had been telling me about having the problem for a long time, but I wasn't bothered by it, and hadn't ever gotten into getting it to happen so I could find the reason. Albert said that it had something to do with the number of backslash commands embedded in the file. That rang a bell concerning the variable SPECIAL in JUST. Special contains a count of the control characters embedded in each line of text, so that if and when that line is justified, a correction can be made for the extra characters that will not print. Well, JUST handled the embedded printer control fine in paragraphs that were formatted or justified, since the procedure fix_end reset the count to zero for the next line, or rather to the number of control characters remaining in the unprinted tail of the last line, which is moved to the beginning of the linebuffer for the next line.

The problem was in the printing with no justification or fill. Special was NOT reset under those conditions, which for a while was perfectly OK since it is not used if paragraphs are not filled, but eventually Special, which was a byte variable, overflowed, and became -128. Since Special is added to the line length in several places in the program, the effect was to make line length negative. Now outputting characters until CUR_LENGTH > LIN_LEN, resulted in one character per line. The cure is simple if you have the PL9 version and have PL9, or if you have the OS-9 "C" version and have the "C" compiler. Down near the end of procedure main is the code in listing A. Note the two starred comment lines containing the statement "SPECIAL = 0;". Just add this statement at that point in the program and recompile, and the bug is cured, and thanks Albert, for finding that one for me. The listing is from the PL/9 6809 code version, but the "C" version is almost line for line with this copy, except that main() in the "C" version is not at the bottom of the program, but just after all the variable and constant declarations.

Editor's Note: The Mustang-08/A uses the same CPU card, only we have already worked out all the problems encountered in assembling a complete system. However, if any of you are having problems getting the board up and running using your own parts, the support from us or your supplier should be able to help you get things running o.k. Even if you bought it from someone else, we will offer what help we can to assist you in getting things right. After all, that is what it is really all about, helping each other.

See Mustang-08/A ad, page 7 for additional information. Or if you don't want to fight the hassle, then consider the Mustang-08/A, then figure what you will pay just for the parts alone! Pretty good deal!

DMW

```
/* MAIN PROGRAM LOOP */

   WHILE NOT(EOF(.INFILE))
   BEGIN
      IF CH = '. .AND FIRST_LINE
         THEN DO_COMMAND
         ELSE FIRST_LINE = FALSE;
      GETALINE;
      IF CONSEC > 1 .AND FILL THEN
      BEGIN
         FILL = FALSE;
         PUTALINE; /* DUMP BUFFER */
         BLANKLINE(1); /* FOR SECOND CR */
      END
      IF CUR_LENGTH <> LIN_LEN - INDENT +
SPECIAL
          .AND NOT(EOF(.INFILE))
         THEN BEGIN
           CUR_LENGTH = CUR_LENGTH + 1;
           LINE(CUR_LENGTH) = CH;
           CH = READ (.INFILE);
         END;
      IF CH = '. .AND ENDLIN THEN DO_COMMAND;
      IF CUR_LENGTH = LIN_LEN - INDENT  +
SPECIAL .AND FILL
         THEN BEGIN
            SPACE_COUNT;
            PUTALINE;
            FIX_END;
         END;
      IF NOT(FILL) .AND CUR_LENGTH <>0
          THEN BEGIN
             PUTALINE;
/*******************************/
             SPECIAL=0;
/*******************************/
             CUR_LENGTH = 0;
          END;
   END;
```

EOF

FOR THOSE WHO NEED TO KNOW

68 MICRO JOURNAL™

FORTH

A Tutorial Series

By: R. D. Lurie
9 Linda Street
Leominister, MA 01543

COMMAND LINE ARGUMENTS

FORTH thrives on arguments passed in the command line, provided that the arguments are entered before the command word. Very nearly all of the common FORTH utilities require one or more parameters on the Data Sta k prior to execution. This has been carried over into every-day FORTH programming pra ti , such that a FORTH command line often looks as cryptic as a C command line; the main difference being the order in whi h the various elements are listed.

This practice may be acceptable for most people, but I find it intolerable! The reason is that I easily forget the order and quantity of required parameters for those utilities which I don't often use. This means that I must dig out the documentation from some usually in onvenient location (Murphy's Law in action) and search through it for the information I need. Either that, or I make an estimate of the correct o der and quantity and go ahead with the command. Can you guess whi h one I do most of the time?

Well I have decided to do something about it at last. Any new FORTH programs that I write will allow command line parameters, if that is appropriate, but there will also be included the necessary error checking and prompting, in ase I don't remember all of the stuff a program needs. Eventually, I will get around to adding this same protection to my existing utilities and programs, as the mood strikes me. This will probably be just after I have made some terrible blunder!

A nice thing about FORTH is that I can write one set of appropriate definitions and use it as a screening module for any program I write. Furthermore, I can add this module to any existing FORTH program at any convenient time.

Sin e I think that there must be other people who have the same kind of memory problem that I do, I have decided to share my current effort with you. This is probably not the final form of the module to be set in concrete, but it is workable. I will use it while I work on improvements.

COUNTING THE PARAMETERS

Fortunately, no fan y parsing of the command line is necessary in order to determine if there are enough parameters on the Data Stack. There already is the word DEPTH in FORTH-83 whi h tells us how many 16-bit integers are currently on the Data Stack. If your FORTH doesn't have DEPTH yet, you can use this definition, which I have taken from Wilson Federicci's FF9.

```
: DEPTH SP@ S0 @ SWAP - 2/ ;
```

The components of this definition are common to all of the versions of FORTH that I have ever seen, so you should not have any trouble using it. Simply comparing the number returned by DEPTH with the number of parameters whi h should be on the Data Stack will tell you whether or not you have enough of them.

PARSE-COMMAND-LINE

Screen #27 contains the hea t of the utility. The definition for PARSE-COMMAND-LINE uses the algorithm shown in Figure 1.

```
1.  Count the number of command line parameters and compare
    this count with the expected number of parameters. If
    the count is not correct, then print the help screen, a
    prompt, and abort (the latter two are optional).
2.  If the parameter count is correct, then verify that the
    order and value of each parameter is within reasonable
    expectations.
```

Figure 1. The algorithm for PARSE-COMMAND-LINE

Those of you familiar with C programming will recognize ARGC and ARGV , but ARGV has a different meaning in this context.

The calling program must have already stored the expected command line parameter count into the variable _ARGC before calling PARSE-COMMAND-LINE . Line #1 of screen #27 calls DEPTH. The returned value from DEPTH is compared to the value stored in _ARGC . If the returned value is too small, then help is obviously need, and the routine of lines #2-3 is called. The prompt and ABORT may be left out, if that is an appropriate action, but I cannot think of a situation where you would not be better off starting over with the program, rather than trying to fix the situation with some sort of routine asking for additional input. Go ahead, if you think that I am wrong, but please tell all of us how you processed the needed extra parameters. I think that it would be a programming nightmare.

If line #1 of screen #27 finds that there are suffi ient parameters, the exe ution skips to line #4, which alls ARGV-OK . This routine should do as mu h a possible to establish ARGument Validity. This is easy for programs which require only a few well defined input parameters, but may be difficult for many situations. For obvious reasons, my examples are of the first type.

I doubt that it would be possible to write a generic version for ARGV-OK , be ause of the nearly infinite number of possible combinations of input parameters; the efore, I have given in, in what I hope is a graceful manner, and accepted the inevitable. The example of ARGV-OK shown in screen #26 is for the specific case of two input parameters. Furthermore, the first parameter cannot be larger than the second, though they could be equal.

No matter what you do within ARGV-OK , be sure to save all of the input paramete s before you do anything else. Othe wise, you uld experience an embarrassing crash!

In my opinion, it is not necessary for ARGV-OK to return an argument (a boolean flag), since the only possible a tions are either for the program to accept all of the input as valid, or to call for the help screen and halt. Notice that I

used QUIT , instead of ABORT , in this definition, since I wanted to preserve the contents of the Data Stack. In that way, I could print the stack contents and see where I made the mistake. Just be sure to clear the Data St ck before resuming operation or going on to other things.

I also think that ARGV-OK should appear in the program, even if you don't know of a current need. You can define it this way:

```
: ARGV OK NOOP ;
```

and have the same effect as a NOOP . Later on, if you think of a way to use it profitably, ARGV-OK is already there, just waiting for the expanded definition. At another time, I'll show how to expand this definition without having to recompile the whole program.

The HELP-SCREEN must also be essentially customized for the particular application; however screen #25 does show a generic help screen which could be used for definitions such as QX , QL , INDEX , and CLEAR-DISK .

It may be a little hard to understand what is happening in screen #25 because of all of the quotation marks peppering the screen. The phrase ASCII " EMIT causes a quotation mark to be displayed, and that is the only universal way to do so from within any FORTH output string. Go ahead and substitute if your FORTH offers a specialized way to do the same thing.

Screen #28 shows a very simplified example of an application of PARSE-COMMAND-LINE . In the definition of EXAMPLE, I have assumed that there would be two input parameters, and that the first parameter cannot be larger than the second parameter. This is the case in my previous discussion.

Line #1 stores a value of 2 into the variable _ARGC , because 2 input parameters are expected. Having prepared _ARGC , line #2 now calls PARSE-COMMAND-LINE . PARSE-COMMAND-LINE can be made more general by requiring that the calling program tell it, by means of _ARGC , the number of input parameters to expect. Line #3 contains a NOOP simply because I did not want to confuse the sample program with extraneous lines. Screen #29 shows a real- world application of PARSE-COMMAND-LINE .

CLEAR-DISK

CLEAR-DISK is a useful utility which illustrates a practical application of PARSE-COMMAND-LINE . The purpose of CLEAR-DISK is to clear a range of screens to ASCII spaces. There may be more efficient ways to do this job, but this is the only way I could think of to do it in high-level FORTH which would work with any FORTH that I have ever seen. The only problem would be in the LITERAL value of 1024, which is wrong for Stearns Electronics FORTH, which has 512-byte screens. Adjust the 1024 to fit your situation.

CLEAR-DISK works by checking for the correct number of input parameters, and whether or not they are in the correct order. This question of order is important, because you don't want to go around the "number circle" in FORTH-83 clearing all but the screens identified in the command line!

The CR of line #3 is a sop to my sense of neatness by starting a new display line.

The real work of the utility begins with the DO ... LOOP parameters in line #4. The 1+ adds 1 to the value of the last screen, as is necessary for proper termination of any FORTH DO ... LOOP . The SWAP reverses the order of the sc en numbers, so that they will be in the proper sequence of the last-number before the first number.

In line #5, the phrase I BLOCK loads the screen with that number into RAM and places the address of the screen buffer on top of the Data Stack. The LITERAL number, 1024 , is the count of the number of ASCII spaces written into the screen buffer by BLANK (BLANKS is the proper word in FIG-FORTH and some others).

The UPDATE in line #6 marks the screen buffer as having been changed, so that it will eventually be written back to disk in its present form of nothing but spaces.

The phrase I 4 .R prints the number of the cleared screen to let me know that the computer is still doing something useful. The 4 .R causes the screen numbers to be formatted right justified in 4 columns. This keeps everything neat and orderly on the screen and makes it easy to read at a glance.

LOOP simply closes the DO ... LOOP . FLUSH forces the last screens to be written back to the disk without waiting for other disk activity to happen later. In this way, CLEAR-DISK can be the last activity for the session, and the computer can be turned off, without me having to be concerned about whether or not the last couple of screens were actually cleared on the disk.

```
SCR # 25
 0 \ _ARGC  HELP-SCREEN                                  RDL 06/25/87
 1
 2 VARIABLE  _ARGC        \ expected number of input parameters
 3
 4 : HELP-SCREEN ( — )                                 \ RDL 06/25/87
 5     CR  CR
 6      ." Two input parameters are required:"     CR
 7      ."   1.  first-scr#"    CR
 8      ."   2.  last-scr#"    CR  CR
 9      ." In no case, can the value of "  ASCII  "  EMIT
10      ." first-scr#"  ASCII  "  EMIT    CR
11      ." be greater than the value of "  ASCII  "  EMIT
12      ." last-scr#."  ASCII  "  EMIT   CR ;

SCR # 26
 0 : ARGV-OK      ( n1 n2 — )                          \ RDL 06/25/87
 1     2DUP                        \ duplicate the input parameters
 2     SWAP  <          \ first-scr# cannot be greater than last-scr#
 3       IF    HELP-SCREEN
 4         CR  ." ok"  QUIT
 5       THEN ;
 6
 7 \ ARGV-OK must be here so that PARSE-COMMAND-LINE can be
 8 \ compiled, even though it could be a dummy definition.
 9 \ Normally, this is the place for verifying the validity of the
10 \ input parameters.

SCR # 27
 0 : PARSE-COMMAND-LINE    ( — )                       \ RDL 07/01/87
 1     DEPTH     _ARGC  @     <   \ minimum argument count?
 2       IF    HELP-SCREEN        \ wrong number of arguments
 3         CR  ." ok"  QUIT       \ start over
 4       ELSE  ARGV-OK            \ verify argument validity
 5       THEN ;

SCR # 28
 0 : EXAMPLE      ( first-scr# last-scr# — )           \ RDL 06/25/87
 1     2  _ARGC  !                \ expected no. of input parameters
 2     PARSE-COMMAND-LINE
 3     NOOP ;                     \ any sort of definition is ok

SCR # 29
 0 : CLEAR-DISK  ( first-scr# last-scr# -- )           \ RDL 06/25/87
 1     2  _ARGC  !                \ expected no. of input parameters
 2     PARSE-COMMAND-LINE
 3     CR
 4     1+  SWAP  DO
 5       I  BLOCK    1024    BLANK
 6       UPDATE
 7       I    4  .R
 8     LOOP
 9     FLUSH ;
```

FOR THOSE WHO NEED TO KNOW

68 MICRO JOURNAL™

Logically Speaking

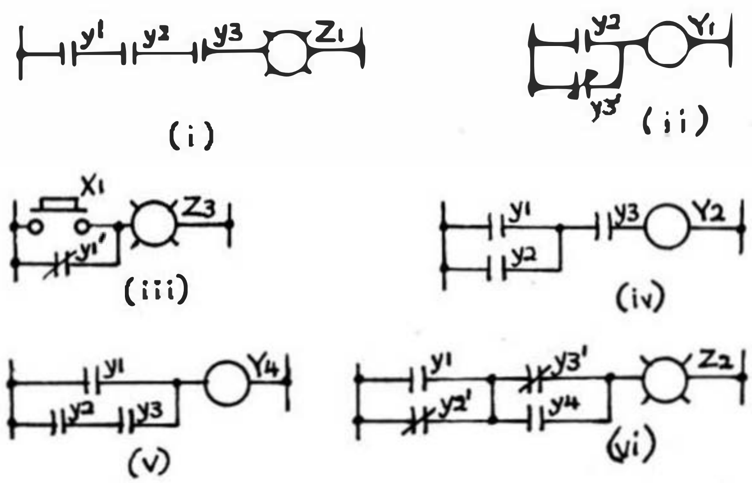
Most of you will remember Bob from his series of letters on XBASIC. If you like it or want more, let Bob or us know. We want to give you - *what you want!*

The Mathematical Design of Digital Control Circuits

By: R. Jones
Micronics Research Corp.
33383 Lynn Ave., Abbotsford, B.C.
Canada V2S 1E2
Copyrighted © by R. Jones & CPI

Solutions to TEST ONE problems :

1.



2. (i) L1 = (y1' + y3)y2' Note the parentheses (hereinafter referred to as "parens") around the y1' and the y3, to indicate that as a pair they are in series with y2'.

(ii) Y1 = y1'y4' + y2y3
(iii) Y2 = y1(y2+y3') + y4'

Don't feel too badly if you didn't get them all right. For many of you this is new and strange, so if you got two or three correct you can feel that you're doing OK. If not, study the solutions and compare them with the original problem to decide where you went wrong. In fact, the early part of our journey will most likely seem the most difficult, but a few miles down the road, when you become more familiar with the language and customs of this new world, you'll find it all remarkably easy. Like learning to ride a bike - it's easy for an accomplished rider to tell you to 'Just hop on and pedal'. But once you've suffered a few bruises, you suddenly find that you can balance OK, and from then on you're off and away. That's not to say that everything after that is easy. Far from it, as you still have to work hard to pedal up a steep hill!

Keep in mind my earlier warning, and don't indicate that you're ready to move on till you feel you've got the hang of TEST ONE! So if we're all fit, here goes for the next stage.

Mile 1 - heading for Mile 2

BASIC LAWS OF BOOLEAN ALGEBRA

Now that we've mastered (more or less) the symbols of Boolean algebra, and have learned how to translate a network into a Boolean expression and back again into a circuit diagram, it only remains to learn eight basic laws and we'll have covered the bulk of Boolean algebra as it applies to switching networks. It's not quite as simple as it might seem at first glance, however. Knowing the rules, and knowing when to apply them are two entirely different matters. Only practice, and more practice, will make a Boolean expert of you!

Later on, though, you'll learn much more powerful and easy-to-use techniques which will take away a lot of the drudge-work. Rather like using a calculator to figure out the square-root of PI instead of doing it by hand. But first we must learn to crawl, then to walk before we can run!

Each of the eight laws mentioned above is really a twin as you'll see from the Table below, and it's one of the axioms of Boolean algebra that any law or statement found to be true for a parallel circuit has an equally true "dual" in a series circuit, and vice-versa. Duals are formed by inverting in the given expression all 1s and 0s, and . and +, but not the letters, or "literals" to give them their correct name. For example, the dual of a' + b = 1 is a'b = 0. Note that the NOT (or complement-sign) stayed with the "a". We did not invert the literals to give the incorrect expression ab' = 0.

I know I said in our first session that Relays, etc., are symbolised with "y", but for a little while I'll use the early letters of the alphabet, as it's easier to say "abc" than "y1y2y3".

Here then are the laws, followed by a detailed explanation of each one, as it's more important for you to UNDERSTAND them than to memorise them.

RULE	Parallel Circuit	Series Circuit
1	a + a = a	a.a = a
2	a + a' = 1	a.a' = 0
3	1 + a = 1	0.a = a
4	0 + a = a	1.a = a
5	a + ab = a	a.(a + b) = a
6	a + a'b = a + b	a.(a' + b) = ab
7	ab + ac = a.(b + c)	(a + b).(a + c) = a + bc
8	(a + b)' = a'.b'	(a.b)' = a' + b'

Some of the rules may look familiar to those of you who are conversant with elementary algebra, but you'd be well advised not to be misled by the resemblance. Instead, let's look at them from a "logical", rather than a mathematical, viewpoint, though we cannot entirely avoid SOME math, as we are, after all, learning how to design control-circuits mathematically. I'll interpret these rules for you from two different viewpoints, and you may adopt either one to suit yourself.

First then, let's look at them from a practical circuit viewpoint. One of the easiest ways I know of for a novice to appreciate the truth of these laws is to imagine that we have two boxes, A and B, to represent each side of a given equation. Each box has a light mounted on its top, plus a few switches labelled "a", "b", "c", etc. Internally, we'll wire up A's light according to the terms on the left of the "=" sign in an equation, and B's light according to those on the right of the "=" sign. We cannot see the internal relays or their contacts - all we can see are the switches and the lights on the top of each box. Now, if for all IDENTICAL combinations of switches on both boxes, we find ourselves unable to distinguish which network is which, then we can truly say that the left-hand expression is equal to the right-hand one.

Take the case of Rule 1 for a parallel circuit. One network consists of two N.O "a" contacts, in parallel, controlling its light, and the other of a single N.O "a" contact. Obviously if switch-a is OFF on both boxes (and thus the corresponding relay-A) then both lights will be OFF, and if switch-a IS operated they will both be ON. As these are the only two possible conditions in this circuit, then we can say that the law is TRUE. Similarly with its dual in a series circuit. Do you see this? Forget about mathematics - just look at it from a practical viewpoint!

Coming to Rule 2, parallel circuit, we see that one light is connected to a network consisting of a pair of complementary contacts on relay-A in parallel with each other (that is, a NO contact and a NC contact), while Box-B consists of a simple "1", meaning that its light IS lit - permanently! Note that there is no mention of a relay-contact in the right-hand expression here, only a "1" or "ON" condition, the swiches on this Box playing no part in controlloing B's light. We also note that whether Box-A's switch-a is operated or not, its light will also be permanently ON, either through the NC contact if switch-a is NOT operated, or via its NO contact if the switch IS operated. So we can say that Rule 2 is TRUE.

Let's continue looking at the parallel circuit laws. By this time it should be easy to interpret Rule 3, which states, in effect, that a permanent wire (that is, a short-circuit) across a NO contact is indistinguishable from a permanent connection on its own.

In Rule 4, let's imagine that we simply cut through the short-circuit jumper of Rule 3 in Box-A, and wire up Box-B's light to a NO-contact on Relay-A. So Rule 4 states, in words, that an open-circuit in parallel with an "a" contact cannot be distinguished from an "a" contact on its own. That's not too bad after all, is it? And that's half the Rules covered already. Now we come to multiple-relay networks!

In order to check out the truth of Rule 5, after having wired up both Boxes, we must list the state of both Boxes' lights under 4 separate conditions. That is, (i) neither relay-A nor B operated, in which case both Boxes' lights will be OFF, (ii) A alone operated (both lights ON); (iii) B alone operated (both lights OFF), and finally (iv) both relays operated (both lights ON).

At this stage, a light should perhaps be beginning to glow ahead of us, and you should see a practical application for this Rule. After all, if someone designed a circuit according to the left-hand expression, we can apply Rule 5 and not only sve ourselves the cost of Relay-B, but also do the same job with only one contact, instead of the original three. Which means less parts, less wiring, less chance of a breakdown (only one Relay to go wrong now), and a much simpler circuit-diagram, making it MUCH easier to trace a fault if one should occur.

In fact, even though you may not be able to DESIGN circuits yet, you can at least begin simplifying already-existing ones. This is what we'll be doing very shortly. Given a Boolean expression representing a circuit or a problem in logic (which, after all, is what Boolean algebra was created for) we can carry out simplification with mathematical precision.

To continue, we should interpret these rules in a very general way. That is, we should read Rule 5 as saying that if any circuit-block of literals (represented by "a") forms part of a larger block (represented by "ab"), then the larger block can be eliminated without further ado. From this we conclude that the expression a'bc + a'bcd'e, for instance, is equivalent to a'bc alone (because "a'bc", IN ITS ENTIRETY, forms part of "a'bcd'e"). It would really be an interesting exercise for you to check this out on your Light-Boxes! Another way of looking at this would be to say "To make things easier, let's temporarily replace "a'bc" with "x" to give us "x + xd'e". Now we can eliminate "xd'e", leaving only "x", or "a'bc". Got it?

Rule 6 in words says "If there is a block of literals whose COMPLEMENT forms part of a larger block, then the complement alone disappears from the larger block". From now on, instead of "block of literals" we'll use the correct word "term", and at the same time learn two new words.

If a product term (the significance of "product" will become apparent in the next section), or term whose literals are connected by "." or "AND", contains a literal for EVERY relay in the control system, it is called a "minterm". On the other hand, if the literals form a sum term (that is, they are all connected by "+" or "OR") then it's called a "maxterm". For example, in a system which has 3 relays, A, B and C, a'bc is a minterm, but "a" is not because not ALL the relays in the system are included - thus "a" is simply a "term". Similarly, a + b + c, or (a + b + c), is a maxterm, but (a + c) is not.

Rule 7 says "If any term forms part of two distinct terms it may be written down only once (or "factored out" to be precise), and then, inside a set of parens, is written down all that remains after deleting this common factor." Thus in the series expression for Rule 7, the term "a." is common to both terms "ab" and "ac", so we write down "a." followed by "b + c" inside parens, as this is what would be left if the "a." were removed from the two original terms. Aha! Exactly like ordinary algebra, say you math enthusiasts. But watch out for the dual, (a + b).(a + c), which is not at all like ordinary algebra! Here the term "a +" is common to both terms, so we write down "a +", followed by "(b).(c)", or more simply "b.c" inside parens, thus "a + (b.c)", and finally "a + bc".

Rule 8, known as 'de Morgan's theorem', states the rule for negating (i.e. finding the complement, or opposite, of) a term or expression. It may be summarised as :

"Complement everything, whether literals, 1s or 0s, or Boolean symbols."

Take care not to confuse (ab)' with a'b'. a'b' represents two NC-contacts IN SERIES, whereas (ab)' is the complement of "ab", or "a' + b'". Note that in this case the implied "." or "AND" between "a" and "b" must also be complemented, giving us a network of two NC-contacts IN PARALLEL.

Now let's look again at these 8 Rules, without benefit of light-boxes, in order to verify whether they're TRUE or not. To begin with, we should read the "OR" sign "+" as a mathematical addition sign, and the "AND" sign "." as a multiplication sign. Hence "sum terms" and "product terms". Basic Boolean algebra has no subtraction or division! And we'll also replace every NO-contact (on a sheet of scrap-paper) with a "1" and every NC-contact with a "0".

So, in Rule 1, we have 1.1 = 1 and its dual 1 + 1 = 1. Here again "1.1=1" is the same as in ordinary math, but in Boolean algebra (which is restricted to just 0s and 1s) "1 + anything" = 1 (see Rule 3 for this, where "1 + a" = "1" and "a" can stand for any term whatsoever, even another "1"). Unfortunately, the fact that both sides of the equation are equivalent does not necessarily mean that it's TRUE for all combinations. However, if one side evaluates to "1" and the other to "0", then the equation is DEFINITELY FALSE. In the case of Rule 1, therefore, we must try out all other Relay combinations. There is only one other possibility - Relay-A becomes energised, and our "a" contacts reverse their sign, becoming 0.0 = 0 and 0 + 0 = 0 for the dual. Again both sides agree, so the total equivalence is TRUE.

Let's try Rule 5 (series) as a further example. 1.(1 + 1) = 1 reduces to 1.1 = 1 (because "1 + anything" = 1) and thence to 1 = 1, which is obviously TRUE. Now let's try energising Relay-A, to give 0.(0 + 1) = 0, which is TRUE, as "0 . anything" = 0. Or in physical terms, an open-circuit in series with anything is equivalent to an open-circuit. Similarly, if Relay-B alone is operated, we have 1.(1 + 0) = 1, again TRUE because the "1 + 0" inside the parens evaluates to 1, giving 1.1 = 1. And finally, if both Relays are energised, we'll have 0.(0 + 0) = 0. As the equation holds for ALL combinations of Relays, we can finally say that Rule 5 is TRUE.

Let's try Rule 8. (1 + 1)' = 0.0. H'm!! 1 + 1 equals 1, reducing the equation to (1)' = 0. In other words, the complement of 1 = 0, which is certainly TRUE. Again, if you try all combinations of Relays, you'll find the Rule to be consistently TRUE. Just remember :

a - replaced with "1" if the Relay is not operated, and "0" if it is.
a' - replaced with "0" if the Relay is not operated, and "1" if it is.
1 and 0 - remain as is, as a short-circuit remains a short-cicuit, no matter what an adjacent relay-contact is doing, and similarly with a "0", or open-circuit.

A DIFFERENT EXAMPLE OF MINIMISATION USING BOOLEAN ALGEBRA

Let's consider a sentential example for a change of pace, that is, a problem in sentence form, by examining the following set of rules for admission to a Computer Club (which shall remain anonymous) :

"A person will be deemed eligible for membership if he/she meets one or more of the following conditions :-

```
     1.  Is married, and is 30 years of age or over.
OR   2.  Is a male under 30 years of age.
OR   3.  Has 3 sponsors, is married and under 30 years of age.
OR   4.  Has 3 sponsors and is a married male.
OR   5.  Does not have 3 sponsors, but is a married female.
```

On reading these rules, they appear to be straightforward enough, though a little sexist at times, and are so clear in their intent that there doesn't seem to be much we can do with them. This is because (as I've mentioned earlier) words have a tendency to "cloud over" the underlying meaning, so let's see what Boolean algebra can do with this particular problem. It will serve as an excellent example of the application of the laws of Boolean algebra to this type of situation.

Our first task is to seek out and list the basic phrases, or "variables", in the five conditions set out above, as shown in the table

```
VARIABLES :  Let        male = m              female = m'
                     married = x         not married = x'
                    under 30 = y          30 or over = y'
                  3 sponsors = z      not 3 sponsors = z'
```

That part wasn't too difficult, was it? Next we'll translate the Rules of Membership, and say that a person will be eligible if :

$xy' + my + xyz + mxz + m'xz'$

How about that? Now we can treat it on a purely mathematical basis, without any emotional hang-ups over age or sex or sponsors, and see what we can come up with. First we'll "factor out" the "x" in terms 1 and 3, which transforms $xy' + xyz$ into $x(y' + yz)$, and apply Boolean algebra's Rule 6 to the contents of the brackets to give us $x(y' + z)$. Remember that if a term appears in an expression and its complement forms part of a larger term, then the complement disappears from the larger term. Then we'll apply Rule 7 backwards to produce the expanded form $xy' + xz$. Putting these in place of the original terms 1 and 3, we have :

$xy' + my + xz + mxz + m'xz'$ which is a slight reduction.

Now for a bigger one. We'll apply Rule 5 to terms 3 and 4 of our reduced expression. This Rule states that if a term in its entirety forms part of a larger term, then the larger term disappears. So out goes mxz to give :

$xy' + my + xz + m'xz'$ That's a LOT better!

But we haven't finished yet. Let's take our new terms 3 and 4 and factor out the "x" to give $x(z + m'z')$, and examine the terms inside the parens. Here again, by applying Rule 6, we can reduce to $x(z + m')$, and just as we did in the first stage of reduction, we'll expand back out to $xz + xm'$. Now we have :

$xy' + xz + xm' + my$ Another slight improvement.

Now for a VERY sneaky operation! First, we'll factor terms 3 and 1 (in that order) to produce $x(m' + y')$. Observe now that term 4 is a term whose complement $(m' + y')$ forms part of a larger term $x(m' + y')$, so that the complement can disappear from the larger term (Rule 6), thereby reducing it to a simple "x". This means that the whole of the original expression has now been reduced to :

$x + my + xz$ Now we're getting somewhere!

but we STILL haven't finished, as we notice that term 1 forms in its entirety part of term 3, which means that the whole of term 3 can be eliminated, reducing us finally to :

$x + my$

Quite obviously, there is no possibility of any further reduction here, so we can translate this expression back into verbal terms as :

"A person is eligible for membership if married or is a male under 30 years of age."

What an enormous reduction this represents, and there is no doubt that without this mathematical approach it would be EXTREMELY difficult, if not IMPOSSIBLE, to arrive at such a solution. Looks like unmarried females and unmarried males 30 or over are O-U-T. I wonder why?

If nothing else, this example should have demonstrated to you that familiarity with the Rules of Boolean algebra does not of itself solve this kind of problem for you. You must acquire the skill to study a given expression and to be able to say, "Aha! Here is where I'll apply Rule 7, then it will be possible to apply Rule 5 dual, and then .... ". So you see, the ability to manipulate or simplify Boolean expressions algebraically (at this stage, anyway, until you learn much more powerful, much simpler techniques later) depends almost entirely on the recognition of the laws of Boolean algebra as they appear in their various forms in the problem involved.

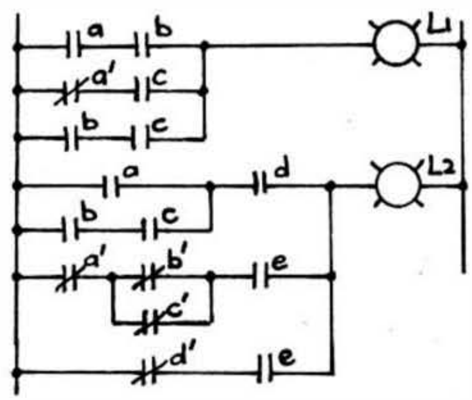
COULD THIS POSSIBLY BE TRUE?

It has been suggested (I forget where) that the legal profession, and our lawmakers in government, actually apply Boolean algebra in reverse. That is, they begin with a simple statement of whatever it is they wish to implement, then expand backwards, retaining their original intent, but bringing in more and more extraneous material (such as the "sponsors" in our example above - which turned out to be totally unnecessary), and only when the whole mess is absolutely incomprehensible - even to themselves - do they finally pass it into law. For myself, I don't believe this for one second! What a suggestion to make!! No, I think they have a natural gift for producing verbal obfuscations (look that one up!), and if they knew that you and I can now apply Boolean algebra to find out what they're REALLY saying, they might even pass another law making it a capital offence for us to do so, otherwise we might even discover that they're saying nothing of importance at all!

BACK ON TRACK AGAIN

To get back to more serious things, and before we come to TEST TWO, and the end of Mile 1, I'd like to show you how to use the same sort of technique to simplify a more complicated relay network than we've come across so far.

EXAMPLE Simplify the following network, if possible :



The first step, as in our previous example for membership, is to derive the logic expressions for the two lights in the network. These are :

L1 = ab + a'c + bc
L2 = (a + bc)d + a'(b' + c')e + d'e

Don't just take my word for it. ALWAYS check it out for yourself. Who knows, you may come across a typo or two on my part! If so, please let me know.

OK, let's tackle the simpler L1 expression first, and begin by applying Rule 4 to term 3, thus :

L1 = ab + a'c + 1.bc

Then we'll apply Rule 2, and replace the "1" with (a + a'). "Hey!", you're going to say, "this looks more like expansion than simplification!". But bear with me for a while - sometimes we have to expand before we can contract. Anyway we now have :

L1 = ab + a'c + (a + a')bc which (by Rule 7) expands to :

L1 = ab + a'c + abc + a'bc

With a double application of Rule 5 to terms 1 and 3, and terms 2 and 4, where in each case a term in its entirety forms part of a larger term, the larger terms can disappear, to produce a result of :

L1 = ab + a'c   where we grind to a halt.

But that's not TOO bad - after all, we got a reduction of 33% in our contacts. So now we're ready to tackle Light-2.

The first thing we notice is that terms 2 and 3 have an "e" in common, which we'll factor out, to give :

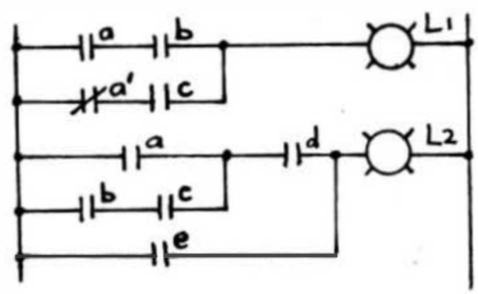
L2 = (a + bc)d + [a'(b' + c') + d].e   Check this out!

And then by a stroke of genius we notice that [a'(b' + c') + d] is the complement of term 1, which means, according to Rule 6, that the complement can disappear from the larger term, to give :

L2 = (a + bc)d + e

Actually it wasn't a stroke of genius at all! In reality, I wrote the expression for Light-2 so that it JUST happened to come out that way, but it DOES serve to demonstrate the technique.

The resultant network is shown in Diagram 5b below, which shows that the net result of applying Boolean algebra to the two Lights' networks is a saving of 7 contacts out of an original 16 - almost 50%. More importantly, the end circuit is going to be more reliable than the original, and certainly much easier to trouble-shoot in the event of circuit malfunction.



Did we all manage to get through that bit of a swamp, or is someone still bogged down back there? If so, just relax a little, re-read the text, and take another run at it. If you still find yourself floundering, don't just thrash around. Come up for air, and take it one step at a time! Soon you'll be learning a new system which will AUTOMATICALLY apply all the correct Rules of Boolean algebra to an expression and enable you in one step to go directly from the original expression to the finally simplified one. But don't neglect to study all that I've just been explaining to you on that account ... you'll still find a knowledge of Boolean algebra a VERY useful skill to have at your finger-tips. Anyway, shortly after learning the new system, and how to use it to simplify expressions, we'll be designing our first simple control-circuits directly from a set of verbal or written instructions.

Here's a simpler example of reduction, for a little light relief :

EXAMPLE   Simplify the following expression

(a + b)(a + b + c + d)

We'll do this one three different ways, believe it or not. Firstly, we can apply Rule 5 and say "Here we have a term (a + b) which in its entirety forms part of a larger term (a + b + c + d), so the larger term can disappear, to leave only (a + b), or a + b." Secondly, we could say "We'll factor out the common (a + b +) to give a + b + 0.(c + d), which again gives us simply a + b. Note that in this case we have to visualise (a + b) as being (a + b + 0), to leave us with 0 when we remove the common factor (a + b +). Thirdly, and you should keep this technique in mind, we can say "Let's take the dual of the expression, to give (remember the rules for forming a dual?) ab + abcd. Now we can "disappear" the term abcd, leaving only ab, and then dual it back again to its original form to produce a + b."

New Boolean usages coming up - the original expression above is called a "product of sums" expression (keep in mind the implied ".", or multiplication, between the two sets of parens) and its dual "ab + abcd" is called a "sum of products" expression. Just note this casually in passing!

Continued on page 37

Telephone: (615) 842-4600

South East Media

Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

SOFTWARE

!!! Please Specify Your Operating System and Disk Size !!!

SCULPTOR

Full OEM & Dealer Discounts Available!

THE SCULPTOR SYSTEM

Sculptor combines a powerful fourth generation language with an efficient database management system. Programmers currently using traditional languages such as Basic and Cobol will be amazed at what Sculptor does to their productivity. With Sculptor you'll find that what used to take a week can be achieved in just a few hours.

AN ESTABLISHED LEADER

Sculptor was developed by professionals who needed a software development tool with capabilities that were not available in the software market. It was launched in 1981 and since then, with feedback from an ever-increasing customer base, Sculptor has been refined and enhanced to become one of the most adaptable, fast, and above all reliable systems on the market today.

SYSTEM INDEPENDENCE

Sculptor is available on many different machines and for most operating systems, including MS DOS, Unix/Xenix and VMS. The extensive list of supported hardware ranges from small personal computers, through multi-user micros up to large minis and mainframes. Sculptor is constantly being ported to new systems.

APPLICATION PORTABILITY

Mobility of software between different environments is one of Sculptor's major advantages. You can develop applications on a stand-alone PC and – without any alterations to the programs – run them on a large multi-user system. For software writers this means that their products can reach a wider marketplace than ever before. It is this system portability, together with high-speed development, that makes Sculptor so appealing to value added resellers, hardware manufacturers and software developers of all kinds.

SPEED AND EFFICIENCY

Sculptor uses a fast and proven indexing technique which provides instant retrieval of data from even the largest of files. Sculptor's fourth generation language is compiled to a compact intermediate code which executes with impressive speed.

INTERNATIONALLY ACCEPTED

By using a simple configuration utility, Sculptor can present information in the language and format that you require. This makes it an ideal product for software development almost anywhere in the world. Australasia, the Americas and Europe – Sculptor is already at work in over 20 countries.

THE PACKAGE

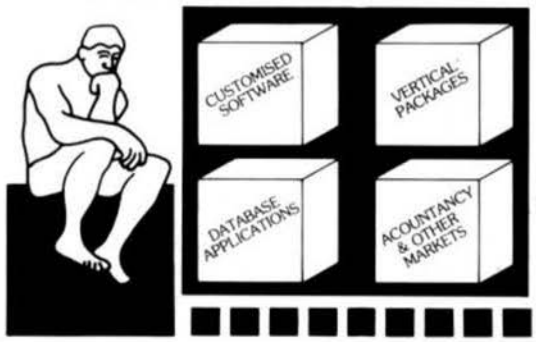
With every development system you receive:

- ☐ A manual that makes sense
- ☐ A periodic newsletter
- ☐ Screen form language
- ☐ Report generator
- ☐ Menu system
- ☐ Query facility
- ☐ Set of utility programs
- ☐ Sample programs

For resale products, the run-time system is available at a nominal cost.

Facts **Features**

Sculptor for 68020
OS-9 & UniFLEX
$995



DATA DICTIONARY

Each file may have one or more record types described. Fields may have a name, heading, type, size, format and validation list. Field type may be chosen from:

- ☐ alphanumeric
- ☐ integer
- ☐ floating point
- ☐ money
- ☐ date

DATA FILE STRUCTURE

- ☐ Packed, fixed-length records
- ☐ Money stored in lower currency unit
- ☐ Dates stored as integer day numbers

INDEXING TECHNIQUE

Sculptor maintains a B-tree index for each data file. Program logic allows any numbers of alternative indexes to be coded into one other file.

INPUT DATA VALIDATION

Input data may be validated at three levels:

- ☐ automatic by field type
- ☐ validation list in data dictionary
- ☐ programmer coded logic

ARITHMETIC OPERATORS

–	Unary minus
*	Multiplication
/	Division
%	Remainder
+	Addition
–	Subtraction

RELATIONAL OPERATORS

=	Equal to
<	Less than
>	Greater than
< =	Less than or equal to
> =	Greater than or equal to
< >	Not equal to
and	Logical and
or	Logical or
ct	Contains
bw	Begins with

SPECIAL FEATURES

- ☐ Full date arithmetic
- ☐ Echo suppression for passwords
- ☐ Terminal and printer independence
- ☐ Parameter passing to sub-programs
- ☐ User definable date format

MAXIMA AND MINIMA

Minimum key length	1 byte
Maximum key length	160 bytes
Minimum record length	3 bytes
Maximum record length	32767 bytes
Maximum fields per record	32767
Maximum records per file	16 million
Maximum files per program	16
Maximum open files	Operating system limit

PROGRAMS

- ☐ Define record layout
- ☐ Create new indexed file
- ☐ Generate standard screen-form program
- ☐ Generate standard report program
- ☐ Compile screen-form program
- ☐ Compile report program
- ☐ Screen-form program interpreter
- ☐ Report program interpreter
- ☐ Menu interpreter

- ☐ Query facility
- ☐ Reformat file
- ☐ Check file integrity
- ☐ Rebuild index
- ☐ Alter language and date format
- ☐ Setup terminal characteristics
- ☐ Setup printer characteristics

SCREEN-FORM LANGUAGE

- ☐ Programmer defined options and logic
- ☐ Multiple files open in one program
- ☐ Default or programmer processing of exception conditions
- ☐ Powerful verbs for input, display and file access
- ☐ Simultaneous display of multiple records
- ☐ Facility to call sub-programs and operating system commands
- ☐ Conditional statements
- ☐ Subroutines
- ☐ Independent of terminal type

MUSTANG-020 Users - Ask For Your Special Discount!

* Tandy CoCo III Special - Reg. $595 * Special $389 *

	*	**	***		*	**	***
MUSTANG-020	$995	$199	$595	PC/XT/AT MSDOS	$595	$119	$595
OS/9 UniFLEX 6809	"	"	"	AT&T 3B1 UNIX	"	"	"
IBM Compatibles	"	"	"	SWTPC 68010 UniF	$1595	$319	$797
Tandy CoCo III	Special $389.00			SWTPC 68010 UNIX	$1990	$398	$995

... Sculptor Will Run On Over 100 Other Types of Machines ...

... Call for Pricing ...

!!! Please Specify Your Make of Computer and Operating System !!!

* Full Developement Package
** Run Time Only
*** C Key File Library

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CC0 = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media

5900 Cassandra Smith Rd. - Hixson, Tn. 37343
Telephone: (615) 842-4600 Telex: 5106006630

master card

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

DISASSEMBLERS

SUPER SLEUTH from Computer Systems Consultants Interactive Disassembler; extremely *POWERFUL!* Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems.

Color Computer SS-50 Bus (all w/ A.L. Source)
CCD (32K Req'd) Obj. Only $49.00
F, S, $99.00 - CCF, Obj. Only $50.00 U, $100.00
CCF, w/Source $99.00 O, $101.00
CCO, Obj. Only $50.00
OS9 68K Obj. $100.00 w/Source $200.00

DYNAMITE+ — Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options w/ OS-9 Version.

CCF, Obj. Only $100.00 - CCO, Obj. $ 59.95
F, S, " " $100.00 - O, object only $150.00
U, " " $300.00

PROGRAMMING LANGUAGES

PL/9 from Windrush Micro Systems -- By Graham Trott. A combination Editor Compiler Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC. 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including interrupts. Comprehensive library support; simple Machine Code interface; step-by-step tracer for instant debugging. 500+ page Manual with tutorial guide.

F, S, CCF - $198.00

PASC from S.E. Media - A FLEX9, SK*DOS Compiler with a definite Pascal "flavor". Anyone with a bit of Pascal experience should be able to begin using PASC to good effect in short order. The PASC package comes complete with three sample programs: ED (a syntax or structure editor), EDITOR (a simple, public domain, screen editor) and CHESS (a simple chess program). The PASC package come complete with source (written in PASC) and documentation.

*FLEX, SK*DOS $95.00*

WHIMSICAL from S.E. MEDIA Now supports *Real Numbers*. "Structured Programming" WITHOUT losing the Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, 6-digit Real Numbers, unlimited sized Arrays (vectors only); Interrupt handling; long Variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. Run-Time subroutines inserted as called during compilation. *Normally produces 10% less code than PL/9.*

F, S and CCF - $195.00

KANSAS CITY BASIC from S.E. Media - *Basic for Color Computer OS-9* with many new commands and sub-functions added. A full implementation of the IF-THEN-ELSE logic is included, allowing nesting to 255 levels. Strings are supported and a subset of the usual string functions such as LEFT$, RIGHT$, MID$, STRING$, etc. are included. Variables are dynamically allocated. Also included are additional features such as Peek and Poke. A must for any Color Computer user running OS-9.

CoCo OS-9 $39.95

C Compiler from Windrush Micro Systems by James McCosh. Full C for FLEX, SK*DOS except bit-fields, including an Assembler. *Requires the TSC Relocating Assembler if user desires to implement his own Libraries.*

F, S and CCF - $295.00

C Compiler from Introl -- Full C except Doubles and Bit Fields, streamlined for the 6809. Reliable Compiler; FAST, efficient Code. More UNIX Compatible than most.

*FLEX, SK*DOS, CCF, OS-9 (Level II ONLY), U - $575.00*

PASCAL Compiler from Lucidata — ISO Based P-Code Compiler. Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.

F, S and CCF 5" - $190.00 F, S 8" - $205.00

PASCAL Compiler from OmegaSoft (now Certified Software) -- For the *PROFESSIONAL*; ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Powerful; Flexible. Requires a "Motorola Compatible" Relo. Asmb. and Linking Loader.

F, S and CCF - $425.00 - One Year Maint. $100.00
OS-9 68000 Version - $900.00

KBASIC - from S.E. MEDIA -- A "Native Code" BASIC Compiler which is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, streamlined, Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.

*FLEX, SK*DOS, CCF, OS-9 Compiler /Assembler $99.00*

CRUNCH COBOL from S.E. MEDIA -- Supports large subset of ANSII Level 1 *COBOL* with many of the useful Level 2 features. Full FLEX, SK*DOS File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System. *A very popular product.*

*FLEX, SK*DOS, CCF - $99.95*

FORTH from Stearns Electronics -- A CoCo FORTH Programming Language. Tailored to the CoCo! Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, Trace, etc. Provides CPU Carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!

Color Computer ONLY - $58.95

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media

5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 South East Media Telex: 5106006630
OS-9, UniFLEX, FLEX, SK*DOS

FORTHBUILDER is a stand-alone target compiler (crosscompiler) for producing custom Forth systems and application programs. All of the 83-standard defining words and control structures are recognized by FORTHBUILDER.
FORTHBUILDER is designed to behave as much as possible like a resident Forth interpreter/compiler, so that most of the established techniques for writing Forth code can be used without change.
Like compilers for other languages, FORTHBUILDER can operate in "batch mode".
The compiler recognizes and emulates target names defined by CONSTANT or VARIABLE and is readily extended with "compile-time" definitions to emulate specific target words.
FORTHBUILDER is supplied as an executable command file configured for a specific host system and target processor. Object code produced from the accompanying model source code is royalty-free to licensed users.
F, CCF, S - $99.95

DATABASE ACCOUNTING

XDMS from Westchester Applied Business Systems
FOR 6809 FLEX-SK*DOS(5/8")
Up to 32 groups/fields per record! Up to 12 character filed name! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced forms! Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!
XDMS-IV Data Management System
XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.
POWERFUL COMMANDS!
XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) oriente which allows almost instant implementation of a process design.

SESSION ORIENTED!
XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV
IT'S EASY TO USE!
XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...)
The possibilities are unlimited...
FOR 6809 FLEX-SK*DOS(5/8") $249.95

ASSEMBLERS

ASTRUK09 from S.E. Media -- A "Structured Assembler for the 6809" which requires the TSC Macro Assembler.
F, S, CCF - $99.95
Macro Assembler for TSC -- The FLEX, SK*DOS STANDARD Assembler.
Special -- CCF $35.00; F, S $50.00
OSM Extended 6809 Macro Assembler from Lloyd I/O. -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. GeneOrate OS-9 Memory modules under FLEX, SK*DOS.
*FLEX, SK*DOS, CCF, OS-9 $99.00*
Relocating Assembler/Linking Loader from TSC. -- Use with many of the C and Pascal Compilers.
F, S, CCF $150.00
MACE, by Graham Trott from Windrush Micro Systems -- Co-Resident Editor and Assembler; fast interactive A.L. Programming for small to medium-sized Programs.
F, S, CCF - $75.00
XMACE -- MACE w/Cross Assembler for 6800/1/2/3/8
F, S, CCF - $98.00

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

UTILITIES

Basic09 XRef from S.E. Media -- This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.
O & CCO obj. only -- $39.95; w/ Source - $79.95

BTree Routines - Complete set of routines to allow simple implementation of keyed files - *for your programs* - running under Basic09. A real time saver and should be a part of every serious programmers tool-box.
O & CCO obj. only - $89.95

Lucidata PASCAL UTILITIES (Requires Pascal ver 3)

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE -- Include other Files in a Source Text, including Binary - unlimited nesting.

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.
F, S, CCF --- EACH 5" - $40.00, 8" - $50.00

DUB from S.E. Media -- A **UniFLEX BASIC decompiler** Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works w/ ALL Versions of 6809 UniFLEX basic.
U - $219.95

LOW COST PROGRAM KITS from Southeast Media The following kits are available for FLEX, SK*DOS on either 5" or 8" Disk.

1. **BASIC TOOL-CHEST $29.95**
   BLISTER.CMD: pretty printer
   LINEXREF.BAS: line cross-referencer
   REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS: remove superfluous code
   STRIP.BAS: superfluous line-numbers stripper
2. **FLEX, SK*DOS UTILITIES KIT $39.99**
   CATS. CMD: alphabetically-sorted directory listing
   CATD.CMD: date-sorted directory listing
   COPYSORT.CMD: file copy, alphabetically
   COPYDATE.CMD: file copy, by date-order
   FILEDATE.CMD: change file creation date
   INFO.CMD (& INFOGMX.CMD): tells disk attributes & contents
   RELINK.CMD (& RELINK82): re-orders fragmented free chain
   RESQ.CMD: undeletes (recovers) a deleted file
   SECTORS.CMD: show sector order in free chain
   XL.CMD: super text lister
3. **ASSEMBLERS/DISASSEMBLERS UTILITIES $39.95**
   LINEFEED.CMD: 'modularise' disassembler output
   MATH.CMD: decimal, hex, binary, octal conversions & tables
   SKIP.CMD: column stripper
4. **WORD - PROCESSOR SUPPORT UTILITIES $49.95**
   FULLSTOP.CMD: checks for capitalization
   BSTYCIT.BAS (.BAC): Stylo to dot-matrix printerr
   NECPRINT.CMD: Stylo to dot-matrix printer filter code
5. **UTILITIES FOR INDEXING $49.95**
   MENU.BAS: selects required program from list below
   INDEX.BAC: word index
   PHRASES.BAC: phrase index
   CONTENT.BAC: table of contents
   INDXSORT.BAC: fast alphabetic sort routine
   FORMATER.BAC: produces a 2-column formatted index
   APPEND.BAC: append any number of files
   CHAR.BIN: line reader

BASIC09 TOOLS consist of 21 subroutines for Basic09. 6 were written in C Language and the remainder in assembly. All the routines are compiled down to native machine code which makes them fast and compact.

1. **CFILL** -- fills a string with characters
2. **DPEEK** -- Double peek
3. **DPOKE** -- Double poke
4. **FPOS** -- Current file position
5. **FSIZE** -- File size
6. **FTRIM** -- removes leading spaces from a string
7. **GETPR** -- returns the current process ID
8. **GETOPT** -- gets 32 byte option section
9. **GETUSR** -- gets the user ID
10. **GTIME** -- gets the time
11. **INSERT** -- insert a string into another
12. **LOWER** -- converts a string into lowercase
13. **READY** -- Checks for available input
14. **SETPRIOR** -- changes a process priority
15. **SETUSR** -- changes the user ID
16. **SETOPT** -- set 32 byte option packet
17. **STIME** -- sets the time
18. **SPACE** -- adds spaces to a string
19. **SWAP** -- swaps any two variables
20. **SYSCALL** -- system call
21. **UPPER** -- converts a string to uppercase

For OS-9 - $44.95 - Includes Source Code
See Review in January 1987 issue of 68 Micro Journal

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 South East Media Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

SOFTOOLS

The following programs are included in object form for immediate application. PL/9 source code available for customization.

READ-ME Complete instructions for initial set-up and operation. Can even be printed out with the included text processor.
CONFIG one time system configuration.
CHANGE changes words, characters, etc. globally to any text type file.
CLEANTXT converts text files to standard FLEX, SK*DOS files.
COMMON compare two text files and reports differences.
COMPARE another check file that reports mis-matched lines.
CONCAT similar to FLEX, SK*DOS append but can also list files to screen.
DOCUMENT for PL/9 source files. Very useful in examining parameter passing aspects of procedures.
ECHO echos to either screen or file.
FIND an improve find command with "pattern" matching and wildcards. Very useful.
HEX dumps files in both hex and ASCII.
INCLUDE a file copy program that will accept "includes" of other disk files.
KWIC allows rotating each word, on each line to the beginning. Very useful in a sort program, etc.
LISTDIR a directory listing program. Not super, but better than CAT.
MEMSORT a high-speed text file sorter. Up to 10 fields may be sorted. Very fast. Very useful.
MULTICOL width of page, number of columns may be specified. A MUST!
PAGE similar to LIST but allows for a page header, page width and depth. Adjust for CRT screen or printer as set up by CONFIG. A very smart print driver. Allows printer control commands.
REMOVE a fast file deleter. Careful, no prompts issued. Zap, and its gone!
SCREEN a screen listing utility. Word wraps text to fit screen. Screen depth may be altered at run time.
SORT a super version of MEMSORT. Ascending/descending order, up to 10 keys, case over-ride, sort on nth word and sort on characters if file is small enough, sorts in RAM. If large file, sort is constrained to size of your largest disk capacity.
TPROC a small but nice text formatter. This is a complete formatter and has functions not found in other formatters.
TRANSLIT sorts a file by x keyfields. Checks for duplications. Up to 10 key files may be used.
UNROTATE used with KWIC this program reads an input file and unfolds it a line at a time. If the file has been sorted each word will be presented in sequence.
WC a word count utility. Can count words, characters or lines.

NOTE: this set of utilities consists of 6 5-1/4" disks or 2 8" disks, w/ source (PL9). 3 5-1/4" disks or 1 8" disk w/o source.
Complete set SPECIAL INTRO PRICE:
5-1/4" w/source FLEX - SK*DOS - $129.95
w/o source - $79.95
8" w/source - $79.95 - w/o source $49.95

FULL SCREEN FORMS DISPLAY from Computer Systems Consultants -- TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.
F, S and CCF, U - $25.00, w/ Source - $50.00

SOLVE from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a snap! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No 'blind' debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!
Levels I & II only - OS-9 $69.95

DISK UTILITIES

OS-9 VDisk from S.E. Media -- For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.
Level I OS-9 obj. $79.95; w/ Source $149.95

O-F from S.E. Media -- Written in BASIC09 (with Source), includes: REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX, SK*DOS Format so it can be used normally by FLEX, SK*DOS; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX, SK*DOS Directory, Delete FLEX, SK*DOS Files, Copy both directions, etc. FLEX, SK*DOS users use the special disk just like any other FLEX, SK*DOS disk
O - 6809/68000 $79.95

LSORT from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and errors messages.
OS-9 $85.00

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CC9 = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** *Telex: 5106006630*

*OS-9, UniFLEX, FLEX, SK*DOS*

HIER from S.E. Media - *HIER is a modern hierarchal storage system for users under FLEX, SK*DOS.* It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. Using HIER a regular (any) FLEX, SK*DOS disk (8 - 5 - hard disk) can have sub directories. By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modem day solution that all current large systems use. Each directory looks to FLEX, SK*DOS **like a regular file, except they have the extension '.DIR'.** A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX, SK*DOS. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!

*FLEX - SK*DOS $79.95*

COPYMULT from S.E. Media — Copy LARGE Disks to several smaller disks. FLEX, SK*DOS utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply i serting diskettes as requested by COPYMULT. No fooli g with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.

Completely documented Assembly Language Source files included.

*ALL 4 Programs (FLEX, SK*DOS, 8" or 5") $99.50*

COPYCAT from Lucidata — *Pascal NOT required.* Allows readi g TSC Mini-FLEX, SK*DOS, SSB DOS68, and Digital Research CP/M Disks while operating under SK*DOS, FLEX1.O, FLEX 2.O, or FLEX 9.O with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishi g a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.

F, S and CCF 5" - $50.00 F, S 8" - $65.00

VIRTUAL TERMINAL from S.E. Media - Allows one terminal to do the work of several. The user may start as many as eight task on one terminal, under *VIRTUAL TERMINAL* and switch back a d forth between task at will. No need to exit each one; just jump back and forth. Complete with configuration program. The best way to keep up with those background programs.

O & CCO - obj. only - $49.95

FLEX, SK*DOS DISK UTILITIES from Computer Systems Consultants -- Eight (8) different Assembly Language (w/ Source Code) FLEX, SK*DOS Utilities for every FLEX, SK*DOS Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). -- PLUS -- Ten XBASIC Programs including: A BASIC Resequencer with EXTRAs over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, a d PRECOMPILER BASIC Programs.

ALL Utilities include Source2 (either BASIC or A.L. Source Code).

F, S and CCF - $50.00

BASIC Utilities ONLY for UniFLEX -- $30.00

COMMUNICATIONS

CMODEM Telecommunications Program from Computer Systems Consultants, Inc. -- Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".

*FLEX, SK*DOS, CCF, OS-9, UniFLEX, 68000 & 6809 with Source $100.00 - without Source $50.00*

X-TALK from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gimix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O Db25 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eight inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk with C-MODEM (68020). Text and binary fi es may be directly transferred between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.

X-TALK Complete (cable, 2 disks) $99.95

X-TALK Software (2 disks only) $69.95

X-TALK with CMODEM Source $149.95

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media

5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

XDATA from S.E. Media - A COMMUNICATION Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC; Re-Transmits bad blocks, etc.
U - $299.99

EDITORS & WORD PROCESSING

JUST from S.E. Media -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use with PAT or any other editor.

• Now supplied as a two disk set:
Disk #1: JUST2.CMD object file,
*JUST2.TXT PL9 source:FLEX, SK*DOS - CC*
Disk #2: JUSTSC object and source in C:
*FLEX, SK*DOS - OS9 - CC*

The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .ce etc.) Great for your older text files. The C source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p ,u ,y etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!
Disk (1) - PL9 FLEX only- F, S & CCF - $49.95
Disk Set (2) - F, S & CCF & OS9 (C version) - $69.95
OS-9 68K000 complete with Source - $79.95

PAT from S.E. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE™". For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.
*Regular FLEX, SK*DOS $129.50*
• SPECIAL INTRODUCTION OFFER • $79.95
SPECIAL PAT/JUST COMBO (w/source)
*FLEX, SK*DOS $99.95*
OS-9 68K Version $229.00
SPECIAL PAT/JUST COMBO 68K $249.00
Note: *JUST in "C" source available for OS-9*

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassel' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - appx. 14,000 plus of free memory! Extra fine for programming as well as text.
*FLEX, SK*DOS $69.95*

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entires lines, etc. Complete with over 25 different CRT terminal configuration overlays.
*FLEX, CCF, SK*DOS $39.95*

SCREDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!
*6800 or 6809 FLEX, SK*DOS or SSB DOS, OS-9 - $175.00*

SPELLB "Computer Dictionary" from S.E. Media -- OVER 150,000 words! *Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX, SK*DOS UCS).* Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.
F, S and CCF - $129.95

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/SK*DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.
NEW PRICES 6809 CCF and CCO - $99.95,
F, S or O - $179.95, U - $299.95

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** *Telex: 5106006630*

*OS-9, UniFLEX, FLEX, SK*DOS*

STYLO-SPELL from Great Plains Computer Co. -- Fast Computer Dictionary. Complements Stylograph.
NEW PRICES 6809 CCF and CCO - $69.95,
F, S or O - $99.95, U - $149.95

STYLO-MERGE from Great Plains Computer Co. -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.
NEW PRICES 6809 CCF and CCO - $59.95,
F, S or O - $79.95, U - $129.95

STYLO-PAK --- Graph + Spell + Merge Package Deal!!!
F, S or O - $329.95, U - $549.95
O, 68000 $695.00

MISCELLANEOUS

TABULA RASA SPREADSHEET from Computer Systems Consultants -- TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

DYNACALC -- Electronic Spread Sheet for the 6809 and 68000.
F, S, OS-9 and SPECIAL CCF - $200.00, U - $395.00
OS-9 68K - $595.00

FULL SCREEN INVENTORY/MRP from Computer Systems Consultants -- Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

FULL SCREEN MAILING LIST from Computer Systems Consultants -- The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

DIET-TRAC Forecaster from S.E. Media -- An XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate, Protein and Fat food exchanges o each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and calorie plan is determined.
F, S - $59.95, U - $89.95

CROSS ASSEMBLERS

TRUE CROSS ASSEMBLERS from Computer Systems Consultants -- Supports 1802/5, Z-80, 6800/1/2/3/8/11/HC11, 6804, 6805/HC05/146805, 6809/00/01, 6502 family, 8080/5, 8020/1/2/35/C35/39/40/48/C48/49/C49/50/8748/49, 8031/51/8751, and 68000 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text.
*68000 or 6809, FLEX, SK*DOS, CCF, OS-9, UniFLEX*
any object or source each - $50.00
any 3 object or source each - $100.00
Set of ALL object $200.00 - w/source $500.00

XASM Cross Assemblers for FLEX, SK*DOS from S.E. MEDIA -- This set of 6800/1/2/3/5/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar TSC Macro Assembler Command Line and Source Code format, Assembler options, etc., in providing code for the target CPU's.
*Complete set, FLEX, SK*DOS only - $150.00*

CRASMB from LLOYD I/O -- Supports Motorola's, Intel's, Zilog's, and other's CPU syntax for these 8-Bit microprocessors: 6800, 6801, 6303, 6804, 6805, 6809, 6811 (all varieties); 6502, 1802/5, 8048 family, 8051 family, 8080/85, Z8, Z80, and TMS-7000 family. Has MACROS, Local Labels, Label X-REF, Label Length to 30 Chars. Object code formats: Motorola S-Records (text), Intel HEX-Records (text), OS9 (binary), and FLEX, SK*DOS (binary). Written in Assembler ... e.g. Very Fast.

CPU TYPE - Price each:

For:	MOTOROLA	INTEL	OTHER	COMPLETE SET
FLEX9	$150	$150	$150	$399
SK*DOS	$150	$150	$150	$399
OS9/6809	$150	$150	$150	$399
OS9/68K	------	------	------	$432

CRASMB 16.32 from LLOYD I/O -- Supports Motorola's 68000, and has same features as the 8 bit version. OS9/68K Object code Format allows this cross assembler to be used in developing your programs for OS9/68K on your OS9/6809 computer.
*FLEX, SK*DOS, CCF, OS-9/6809 $249.00*

GAMES

RAPIER - 6809 Chess Program from S.E. Media -- Requires FLEX, SK*DOS and Displays on Any Type Terminal. Features: Four levels of play. Swap side. Point scoring system. Two display boards. Change skill level. Solve Checkmate problems in 1-2-3-4 moves. Make move and swap sides. Play white or black. This is one of the strongest CHESS programs running on any microcomputer, *estimated USCF Rating 1600+ (better than most 'club' players at higher levels)*
F, S and CCF - $79.95

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

A LITTLE BACGROUND MATERIAL

By way of a little background, I'm not a professional teacher. At one time I was a Systems Designer for a large electrical manufacturer in Toronto, Ontario, and was heavily involved with automating the plant's production machinery. We also did custom automation for other large companies. We had a staff of about 450 - 500, I think. Anyway, each fall and winter I would conduct, after hours, for the benefit of anyone who wished to attend, a series of in-plant lectures on the mathematical techniques I used to design control-circuitry. Sometimes, too, my company would sponsor a series for groups of engineers from other companies, with the added benefit of workshop sessions, where the "students" would construct logic-circuits to test out their designs. About 11 years ago, having worked for other companies in between whiles, I decided to go independent and form my own company. So here I am!

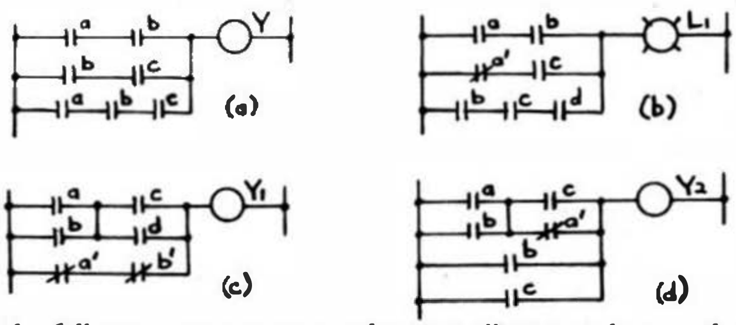
Let me ask you a favour here. Please don't ask me to get involved in any massive project you'll undertake to try out your new-found skills! Believe me, I already have more than enough things I'd like to do to keep me occupied for the next few years, and I don't wish to appear a meanie by having to say NO. Thanks!

TEST TWO

1. Write both the "dual" and the "complement" of the following expressions :

(a) a(b + c') (b) (a + b)(c' + d') (c) ab' + c'd

2. Simplify the following networks, indicating all steps taken, and draw the resultant networks :



3. Simplify the following expressions, indicating all steps taken, and draw both the initial and final networks for comparison :

```
(a)  L3 = x + y + y'
(b)  Y2 = a + bb'
(c)  Y1 = (a + b + c)(d +e)(b + c + d)(a + e)
(d)  L1 = a'b'c + ab'c + a'bc
(e)  a + a'b + a'b'c + a'b'c'd + .....
(f)  (a + bc + e)(a + bc + f)
```

SOME FATHERLY ADVICE

Don't be too discouraged by the apparent complexity of some of these problems. Some of them are real toughies!! Don't forget the new technique that's coming up! Work away at them, and if you manage to get 3 or 4 correct that's a good sign. Remember that you're new to all of this, and you're not going to accomplish miracles overnight. Keep plugging away, and just as with bike-riding, you'll suddenly find that you KNOW how to do it!! Once you know the language, and your muscles (mental ones, that is) have got toughened up a little, this new country won't seem quite so frightening!!

..... *End of File 1. Whew!!!!*

EOF

FOR THOSE WHO NEED TO KNOW

68 MICRO JOURNAL™

EPROM Emulation for the S50 Bus

By:
Robert Lund
789 Kcarnage Elmhurst, Il. 60126
Phone : (312)530-0957
Day phone: (312)574-2000 ext. 588

Have you ever had the need to develop a microprocessor application and found yourself spending more time at the EPROM burner than you first thought possible? Not to mention of course those blinding ultraviolet rays from your eraser? Well, if you have, then just possibly, this project is for you!

For lack of a catchy name, lets just call the device to be described an EPROM emulator. In theory, this device will allow the software developer the ability to assemble (or compile) a program and run it on the target microprocessor system without burning an EPROM. In practice, this emulator consists of 4K of dual ported RAM. One port allows for the host computer to read and write while the other port is accessible by the target system on a read-only basis. In this configuration, the emulator is capable of emulating in RAM either a 2716 or 2732 EPROM. The selection between the two EPROM types is accomplished by a switch on the remote emulator "pod".

SYSTEM DESCRIPTION – In the emulators present configuration, the dual ported RAM on the emulator board (which resides on the host SS-50) begins at address $0000 and extends to address $0FFF. This 4K of RAM replaces the original 4K in the host system. Due to this replacement, the host must have the capability of disabling the RAM that the emulator occupies. In general operation, the RAM on the emulator board appears as normal read/write memory to the host and can be accessed by the host at any time. In the target system, the RAM on the emulation board appears as either 2K or 4K of read-only memory.

You might ask at this point; How can two different computer systems access the same memory at the same time? Well, in reality, they can't. The host system, which is the system that is developing the software (or is it the programmer?) can access the dual ported memory at any time it pleases. The target system can only access the RAM when the host isn't. Oh yes, conflicts are possible. If both the host and the target are contending for the same memory at the same time, someone has to loose. The looser of course is the target microprocessor system. The obvious result in this design is the fact that the target system will go crazy unless it is held in some form of known state. The best resolve to the problem is to hold the target system in a reset mode until the emulation RAM is loaded (we'll get to this shortly).

In all but the most obscure situations, both the operating system and any assembler will be off and in control in higher memory locations. In this situation, the target system will have full read-only access with no possible contention.

CIRCUIT DESCRIPTION – If I have maintained your interest so far, you may have noticed that the actual hardware concepts for the emulator are quite straight forward. There are no hard-to-find components and all bus timing signals are well within system tolerances.

The complete EPROM emulator consists of four circuit functions: Bus interfaces to the SS-50 bus, 4K of low power RAM, data selectors/ multiplexers and line drivers and receivers for the umbilical emulation pod. As mentioned above, all of the components in the parts list can be purchased in small quantities from several mail order suppliers. This includes the byte wide RAM parts, which by the way, have plummeted in cost over the last year. So much so in fact that a full 32K of memory can now be purchased on a single 28 pin part for under $30.00.

Continuing with the circuit description, the first aspect to discuss consists of the SS-50 bus interface. This is where the emulator meets the host. Referring to the schematic diagram, the eight data lines from the host system are buffered by IC1, a 74LS245 tri-state octal bus transceiver. The buffered data is then passed on to the two 6116, 2K X 8 byte memory devices, IC4 and IC5. IC3 is an octal tri-state line driver which buffers the data lines towards the target emulation pod. This particular device provides data buffering in only one direction due to the fact that the target is only reading the data and not writing.

The actual magic which is performed to enable the 6116 memory devices to function as

dual ported RAM is provided by the 74LS157 quad 2-line to 1-line data selectors/multiplexers, IC6,7,8 and IC9. These devices act in a single-pole-double-throw fashion to switch the address lines between the host and target systems. When the host requests the memory through the "board select" line, IC6,7,8 and 9 switch and apply the address lines from the host system to the memory devices. In addition, activation of the "board select" line disables data bus driver IC3. In effect, the memory is totally removed from the target system and is only available to the host. When the "board select" line from the host system is not active, driver IC3 and data selectors IC6,7,8,9 are switched in such a manner as to allow the target system full read-only access to the 6116 memory devices.

IC11,12 and IC13 on the emulation pod or "drive board" simply provide data and address line buffering between the main memory board contained in the host to the emulator pod.

The main board and the pod are interconnected by means of a standard 50 conductor ribbon cable. This cable is similar to the type of cable which is used to connect 8 inch disk drives. The strange portion of the schematic on the left side of the emulation pod drawing is a representation of the bottom side of a standard ribbon cable connector. As noted on the drawing, every other line in the cable is at ground potential for interline shielding. The switch indicated on the emulation pod drawing switches address line A11 which is required to differentiate between the 2716 and 2732 EPROM types.

The board select level is accomplished by 4 line to 16 line decoders, IC14 and IC15. These two ICs provide complete address decoding for the first lowest 4K of address space. My board takes advantage of the 20 bit DAT address scheme and completely decodes all 20 address lines. If you do not need, or can not take advantage of this addressing scheme, simply delete IC14 and only use pin #1 of IC15 as your board select. The green board select and red power LEDs are used simply as visual indicators. Again, if you don't need them, simply delete them.

CONSTRUCTION HINTS — All of the devices on the board select and memory schematics are contained on an SS-50 card. Although it has been found that SS-50 prototyping boards are becoming harder and harder to find, a very effective construction technique can be accomplished by using an appropriate sized 0.1 inch grid glass epoxy Vector board and placing a selection of 10 pin Molex connectors along one edge. If this technique is used, you must provide very low impedance power and ground lines to all of the integrated circuits. This can be done by using number 14 copper wire. It is suggested that the ground be placed on one side of the board and the power lines be placed on the other side. This will provide for easy placement of the necessary IC sockets and liberal placement of 0.01mF. bypass capacitors.

As mentioned earlier, the emulation pod is separated from the main memory board by a three foot long standard 50 conductor cable. Although no problems have been noticed with a cable of this length, it is recommended that the ribbon cable be limited to no more than three feet.

The emulation pod itself was constructed in a small black phenolic box. Entering at one side of this box is the 50 conductor extending to the SS-50/4K RAM board. Protruding from the other side of the emulation pod is a short (less than three inches) cable terminated in a DIP header appropriate for the 2732 EPROM. Also contained within the pod enclosure is the 2716/2732 switch.

Although my emulator was constructed using the above techniques, it is worth mentioning possible problem areas which may crop up during construction and testing. First, and probable the most important, use heavy power and ground lines around the ICs. Also, use at least one 0.01mF. capacitor between power and ground at each IC. Failing to use these two important suggestions will almost insure that eventual problems will occur.

A note of caution is also required in the use of three terminal regulator parts. These devices have a nasty tendency to oscillate if they are not bypassed well. As a general rule of thumb, use a 0.1mf. capacitor on the input side and a 0.1mf. and a 30 to 100Mf. capacitor on the output side of the regulator. The 1N4001 diode between the input and output serves as a protection device. This diode routes the reverse potential around the regulator when the main power on the host is shut off. In effect, this prevents a reverse potential on the regulator between the off or low state power supply and the positively charged bypass capacitors on the board. This precaution will prevent a regulator from mysteriously dying for no known good reason. Also, use an adequate heat sink on this device.

USING THE EMULATOR — Now that you have the emulator constructed, how do you use it? Well, as far as the host system is concerned, the emulator is totally transparent. It acts as any other memory would in the first 4K of memory space. The real key to using the emulator for it's intended purpose is to move any

assembled or compiled code into this 4K space so as to become externally accessible by the target system.

Unfortunately, the above process is easier said than done. Some microprocessors are blessed with a vast complement of relative addressing modes which makes them a breeze to program. The 6809 and the 68000 series are such devices. Other, more archaic designs must rely on absolute addressing modes to accomplish any useful work. The absolute addressing modes of these older processors require that memory locations and various subroutine locations reside at a fixed "absolute" location. If they need a variable at say location $0100, it had better be at location $0100, or else!

Newer processing devices have the capability of addressing memory locations using a "relative" offset from the program counter. The result of this capability is the fact that a program or routine can run in any memory location.

Without going into great detail over these two concepts, save it to say that when the EPROM emulator is used with a microprocessor that requires any absolute memory locations, a form of gymnastics must be applied to the final emulated code.

As you may have already guessed, the emulator memory resides at address space $0000 to $0FFF in the host's address space. This address space may be totally different from the address space occupied or expected by the target system. As an example, if you were developing a software project for say a 6802 microprocessor, the target would expect to see EPROM containing reset and interrupt vectors at $FFF8 to $FFFF. This is drastically different from the emulator address space from $0000 to $0FFF.

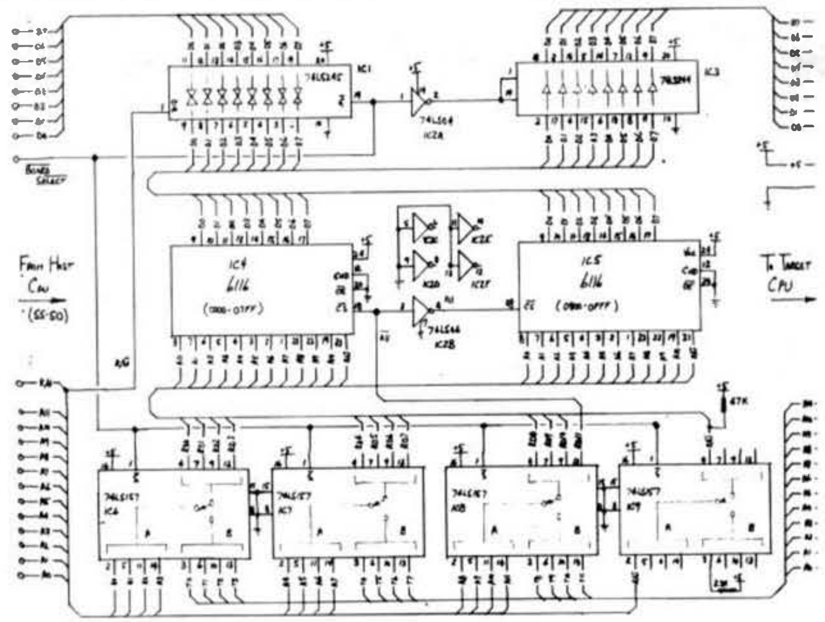
The difficulty results from the fact that attempts by the targeted microprocessor to execute an absolutely addressed memory location will not be correct. As an example, the reset vector on the 6802 is located at $FFFE in it's memory space, not $07FE as would be the case in the emulator/host address space. As such, one must compensate in some way for any absolute memory locations. This can be accomplished in an assembly code program by setting the program's "origin" at $0000 and then telling the assembler to add an offset to any absolute memory references. This can be done by adding an additional offset to any absolute memory referenced instruction similar to this: JMP LABEL would be changed to JMP LABEL+$F800. Once the offsets are added to all absolute references, the addressing scheme will be correct for the target system. To visualize this concept, refer to the following code fragment in Fig. 1:

```
 TTL FIG1.ASM
  PAG
 *********************************************
 * Offsets and code fragment illustrating    *
 * operation for a 2716 EPROM.               *
 *********************************************
  ORG $0000 TARGET'S LOW RAM MEMORY
 TEMP RMB 1

 *********************************************
 * Actual program starts here. Although it   *
 * appears to start at $0000, the target will *
 * see it beginning at $F800. Consequently, all*
 * absolute memory references must include an *
 * offset addition of $F800.                 *
 *                                           *
 * The above addresses starting at $0000 are OK*
 * The target will see them as low RAM.      *
 *                                           *
 * Any absolute addressing in the body of the *
 * following code MUST have $F800 added!     *
 *********************************************
  ORG $0000
 START LDAA #$FF
  STAA TEMP OK HERE, OUT OF EPROM RANGE
  CMPA #$01
  BNE START OK HERE, ADDRESS IS RELATIVE
  JMP START+$F800 OFFSET MUST BE ADDED, IS
ABSOLUTE!

 *********************************************
 * Interrupt and reset vectors               *
 *********************************************
  ORG $07F8
 IRQ FDB START+$F800
 SWI FDB START+$F800
 NMI FDB START+$F800
 RES FDB START+$F800

  END START
```

From the above example, once the program is assembled and moved into the host's memory starting at $0000, all of the absolute memory references will be correct for access and operation on the target microprocessor system.

From the above assembler listing, note that the origin has been set to $0000. If this conscript is followed, the final assembled .BIN file will be loaded at $0000 when using the FLEX "GET" utility.

One of the interesting capabilities of the emulator is the ability to change memory constants in a target system on-the-fly without reassembling the whole program. Remember the last time you tried to tailor that timing loop with just the right constant? Now, by using the emulator, it's a simple matter of accessing the constant's location with the host, inserting a new constant and running the target system.

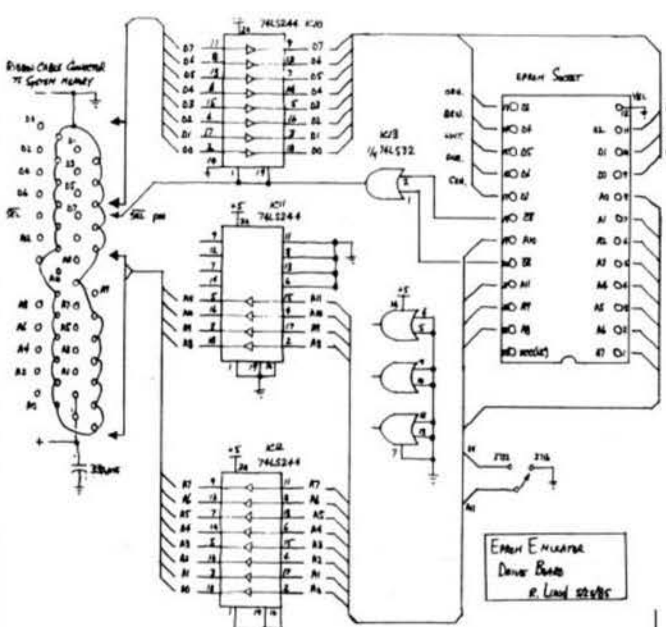


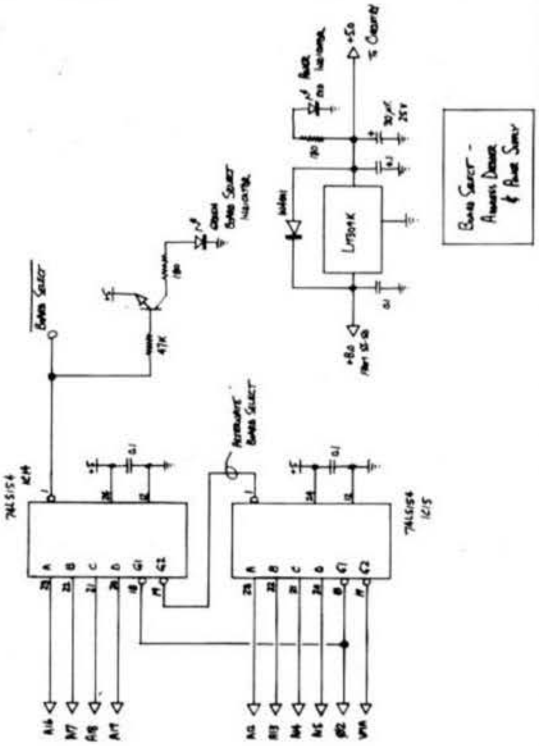
IN CONCLUSION - At this point in time, no printed circuit board has been produced for this project. The emulator was originally designed to be a single device. If there are any ambitious individuals in the audience who would like to partake in board design, be my guest. PC board design is certainly not my cup of tea.

I hope this project is as exciting to you as it has been for me. Once you start using this device, you will quickly find as I did that it is hard to live without.

As a final note, I will answer any questions you may have in regards to this project if you send a postage paid return envelope. Good luck.

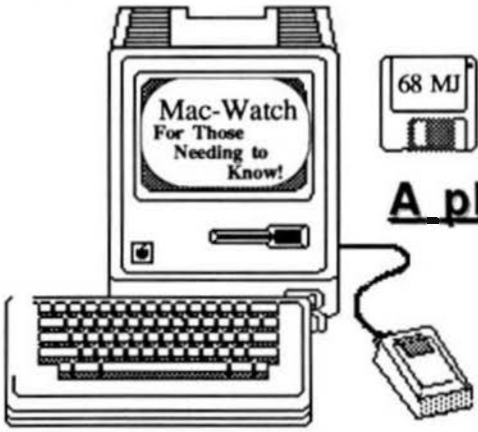
EOF





FOR THOSE WHO ***NEED TO KNOW***

68 MICRO JOURNAL™



The Macintosh™ Section

Reserved as a

A place for your thoughts

And ours.......

Mac-Watch

A REVIEW OF DOUG CLAPP'S WORD TOOLS

By: Ed Law
1806 Rock Bluff Road
Hixson, TN 37343

After considerable delay and several false starts, Doug Clapp's "Word Tools"has been distributed and is on dealers' shelves. According to the distributor, Aegis Development, " . . . Word Tools has been designed to help you write better; to catch mistakes, to suggest better ways to write things, and to encourage you to think about your writing." Let's see how well those objectives are achieved.

Word Tools provides for opening the text document to be analyzed using standard Mac protocols. Word Tools then performs four functions. It counts, sorts, analyzes, and suggests ways to improve text.

T Minus Ten and Counting

Selecting **COUNT** under the **OPTION** menu allows the counting function to be customized. You may get a count of characters, invisible characters, spaces, words, articles, prepositions, proper nouns, and paragraphs. Alternately, you may eliminate those items that do not interest you. After **OPTIONS** have been addressed to your liking, the count is begun by selecting **Count Documents** in the **WORDS** menu or by clicking the count button. If the document being counted is in MacWrite format, get set for a long wait. Documentation with Word Tools quickly admits that MacWrite documents are read slower than ASCII or Word files. If your documents are prepared with MacWrite and you don't have time to spare, save your documents in ASCII format and the problem largely goes away. This 1200 word review took nearly 3 minutes to read while in McWrite format and only 20 seconds to read when saved in ASCII FORMAT.

Upon completion of the counting process, the **RESULTS** window shows not only the number of hits for each item being counted, but also information about averages and extremes. The "Averages"section of the window shows the *average* characters per word; words, prepositions, and articles per sentence; and sentences per paragraph. The "Extremes"section shows the *maximum* characters per word, words per sentence, and sentences per paragraph.

When **RANKINGS** is selected from the windows menu, five bar charts are calculated from the count results and show in the graphic form the documents relative complexity, readability, and interest level. These charts show the relative grade level at which the material is written and its relative word, paragraph, and sentence length. There is no explanation in the users guide about what these charts mean or how they should be used. Also, the program apparently makes no provision for printing these charts in the "Word Tool Report."

The **EXTREMES** window shows the longest paragraphs and longest sentences. (The **OPTIONS** menu allows you to set how many such extremes will be recorded.) As you review each extreme, Word Tools allows you to mark it for future correction or ignore it and move on to the next extreme. This is a useful feature to those who tend toward run-on sentences or paragraphs.

Sorting and Listing Words

As Word Tools counts a document, it produces a list of all words used in that document. The **SORT OPTIONS** window under the **OPTIONS** menu allows the words to be sorted alphabetically, by word length, or by the number of times the word was used. The direction of sorts may be ascending or descending. The resulting tests can be viewed on the screen, printed out in a Word Tools Report, or saved as a separate file. This feature can be used to look for overworked words or to identify overly-complex words that need to be replaced.

Checking Style and Punctuation

One of the major functions of Word Tools is to check for apparent errors in style and punctuation. Since what is right or wrong for a particular usage is highly dependent on the context, Word Tools calls its findings "suspects"rather than errors.

For example, the internal style suspect list warns against using the word "girl" as being potentially offensive. When used in children's literature, however, its use would be entirely appropriate. The importance of this principle cannot be overstated. In a variety of documents checked with Word Tools from excerpts from children's books to technical software reviews, 40-percent of the punctuation suspects and 60-percent of the style suspects were ultimately determined to be satisfactory with no changes needed.

The accuracy of the checking process can be improved by creating your own suspect lists. Word Tools allows you to create as many as you like. Lists may be customized to contain only the types of errors expected depending on the application. For example, a newsman's style suspect list might highlight "Russia"as being an unacceptable reference to the USSR.

When Word Tool is used to check style and punctuation, the resulting suspects may be either ignored or changed. If the suspect is to be changed, the suggestions provided by the suspect list may be used or an alternative approach taken. If the suspect lists' suggestion is to be taken, it can be used to change the specific suspect highlighted or gobally change all occurrences of that suspect in the document. This global change feature may be used as a glossary function. Short codes would be typed whenever long repetitive terms were to be used. These codes would then be entered in a suspect list. Word Tool could then quickly find and replace all such codes with the full term.

Printing Reports

Word Tools can print simple reports showing the results of the word count and the word list. The word list may be sorted in several orders before printing. Unfortunately, the word list printout does not include the frequency of occurrence of each word as is shown on the screen. The excepting lists, ranking bar charts, and other features of Word Tool apparently cannot be printed in this report.

Equipment Requirements

Word Tools works with any Macintosh with at least 512k. Two disk drives are highly recommended. It is not copy protected.

Word Tools Users Guide

The users guide for Word Tools is complete and clearly presented. The Ranking section could have contained more explanation about the meaning of the bar charts and how the ranking of a document can be improved. Overall, the users guide was adequate.

Overall Assessment

Overall, the value to Word Tools to the ordinary Mac user is limited. It does provide interesting statistics about documents which will be of use to linguists and perhaps to professional writers. Its utility would have been greatly increased if it were included in a broader scope program including a high quality spelling checker. The Ranking feature would be more useful if additional rationale were given on how documents are rated as to "interest level"or "grade level." The high number of suspects highlighted in error will quickly discourage many users.

EOF

FOR THOSE WHO NEED TO KNOW

68 MICRO JOURNAL™

Bit-Bucket

By: All of us

"Contribute Nothing - Expect Nothing", DMW '86

Continued From Last Month

XBASIC Xplained

or

Things you won't find in the documentation

Copyright 1986 by
R. Jones, Micronics Research Corp.,
33383 Lynn Avenue, Abbotsford,
British Columbia, CANADA V2S 1E2
& Computer Publishing, Inc. (CPI) © 1987

The material in this article is copyrighted by Micronics Research Corp. & Computer Publishing, Inc. (CPI), and reproduction, in whole or in part, by any means is prohibited without the express written permission of the author and CPI.

And More Ways to Speed & Shorten Programs

Let's start small with Example 1, and work our way up to the not-so-obvious, of which instances appear in all kinds of programs, so consider then :

```
100 IF X% = 1 THEN Y% = 1
110 IF X% = 2 THEN Y% = 2
 .  .   .  .  .
190 IF X% = 10 THEN Y% = 10
200 rest of program
```

I think almost everyone will see my point here, namely that a lot of program-space is eaten up and a lot of time wasted going through all those IF-THENs, every one of which has to be interpreted in turn before Line 200 is reached. I've even seen instances where the author of the program has been aware of the time wasted, and decided to make the program execute faster by tacking the statement ': GOTO 200' on to the end of every line except Line 190. In spite of this, of course, it still means that the time wasted is proportional to the value of 'X%', as it would still have to scan all the way through to line 190 if X% were equal to 10.

How much tidier and faster-to-execute if we simply wrote :

```
100 Y% = X%
200 rest of program
```

which achieves exactly the same thing in a lot less space, and doesn't have to spend time evaluating even one solitary IF-THEN.

All agreed? OK then, let's move on to a much more frequently occurring pattern :

```
100 IF X% = 1 THEN Y% = 17: GOTO 200
110 IF X% = 2 THEN Y% = 20: GOTO 200
120 IF X% = 3 THEN Y% = 23: GOTO 200
 .  .  .  .   .   .  .
190 IF X% = 10 THEN Y% = 44
200 rest of program
```

I should mention that sometimes it might appear in the form 'Y%=17*Z%', 'Y%=20*Z%', etc, but the underlying principle is the same. The main thing to observe is that each time 'X%' increases by 1 then the value associated with Y% increases by 3, ie a ratio of 3:1. This gives us the clue that in some way 'X% * 3' comes into the picture. Let's apply this line of thought to Line 100, where X% = 1 and therefore X% * 3 = 3. Surely this Line could be re-written as :

```
100  Y% = 14 + X% * 3 : GOTO 200
```

with no difference in the result as far as Line 100 alone is concerned. '14' is, of course, the base-constant necessary to make Y% evaluate to 17. Aha! And what's this we find? Nothing less than that our new Line 100 also evaluates correctly for the remaining lines through to Line 190, thus eliminating the need for lines 110 - 190, and also the 'GOTO 200' on the end of Line 100. And so our program reduces to a mere :

```
100  Y% = 14 + X% * 3
200  rest of program
```

Maybe this is so obvious that you are wondering (yawn) why I'm bothering to raise the subject at all, but, believe me, I've seen this very thing so often that I'm convinced some things become obvious only when they're pointed out. So ... let's move on to example 3 :

```
100  IF X% = 1 THEN Y% = 44: GOTO 200
110  IF X% = 2 THEN Y% = 41: GOTO 200
120  IF X% = 3 THEN Y% = 38: GOTO 200
 .  .  .  .  .  .  .
190  IF X% = 10 THEN Y% = 17
200 rest of program
```

Now what do we do? We observe that each time X% increases by 1 then Y% decreases by 3. In this instance although the formula :

```
Y% = 41 + X% * 3
```

holds for Line 100, it certainly doesn't hold for the remaining lines. This too is a very common situation, and one solution is first of all to reverse the expressions for Y% from top to bottom, thus re-creating the situation in example 2. After deriving our formula 'Y%=14+X%*3' for the revised set of lines, we are ready for our second observation, namely that in any line the original value for Y% and the revised value for Y% add up to 61. Thus the original 'Y%=44' got transformed into 'Y%=17', where 44+17=61; the original 'Y%=41' got transformed into 'Y%=20', again adding up to 61, and so on through to Line 190. Another way of saying this is that if we subtract our transformed value from 61, we'll produce the original value. And so our formula 'Y%=14+X%*3' now becomes :

```
100  Y% = 61 - (14 + X% * 3)
                    or
100  Y% = 61 - 14 3- X% * 3
                    and finally
100  Y% = 47 - X% * 3
```

Hey, that's fantastic, as our program can now be reduced to a mere :

```
100  Y% = 47 - X% * 3
200  rest of program
```

But there's a shorter and more direct way to arrive at our final formula! Can you see it? Should I give it to you right now, or wait till next time to give you a chance to figure it out? OK, you've convinced me! Here's how.

Based on our observation that Y% decreases by 3 each time X% increases by 1, let's force a decrease by implementing '-X%*3' instead of '+X%*3', as we did when Y% increased proportionately. Now, let's take a look at Line 100, and see how we can force it to assign the value 44 to Y%, given (-X%*3) which, for this line, equates to -3. Obviously we need the formula 'Y% = 47 - X% * 3' (for Line 100, this is equivalent to 47 - 3, or 44), and lo and behold! the formula holds true for the remaining lines.

If the relationship between one line and its successor doesn't follow a particular pattern, I wouldn't waste time trying to find a reduced program-segment. However, segments of the following form, although the relationship is non-linear, are amenable to solution as there is a pattern of sorts between successive lines. See if you can find a suitable formula!

```
100 IF X% = 1 THEN Y% = 21: GOTO 200
110 IF X% = 2 THEN Y% = 24: GOTO 200
120 IF X% = 3 THEN Y% = 28: GOTO 200
130 IF X% = 4 THEN Y% = 33: GOTO 200
 .  .  .  .  .   .  .
190 IF X% = 10 THEN Y% = 84
200 rest of program
```

Notice that the increment between successive lines isn't constant here, but itself increases by 1 each time. That is, the first increment is 3, the second 4, the third 5, and so on. I'm going to warn you that this one isn't so simple, but, for what it's worth, the solution is given a little further on. Don't peek till you've had a go at it first!!

Now I'm going to give another example, and jump straight through to the solution, leaving you to verify that in fact the greatly shortened program-segment is truly equivalent to the original. Here it is :

```
100 IF X% = 1 THEN Y% = Y%+19*Z%: Z% = Z%+55: GOSUB 1000: GOTO 200
110 IF X% = 2 THEN Y% = Y%+26*Z%: Z% = Z%+51: GOSUB 2000: GOTO 200
120 IF X% = 3 THEN Y% = Y%+33*Z%: Z% = Z%+47: GOSUB 3000: GOTO 200
    and so on down to
190 IF X% = 10 THEN Y% = Y%+82*Z%: Z% = Z%+19: GOSUB 10000
200 rest of program
```

That's some program! But, using our new technique, it all boils down to :

```
100 Y%=Y% + (12 + 7 * X%) * Z%: Z% = Z% + 59 - 4 * X%
190 ON X% GOSUB 1000,2000,3000, .... 10000
200 rest of program
```

Notice how we adapt to the fact that each line goes to a different subroutine before popping down to Line 200, by using ON - GOSUB. It's lines such as Line 190, where I've sometimes had as many as 40 or more addresses to which to 'GOSUB', that make me seriously consider adding a 'computed GOTO/GOSUB' feature to my new RBASIC. In such an event the GOSUB address would be computed before being executed and the program would reduce further to :

```
100 Y%=Y%+(12+X%*7)*Z%: Z%=Z%+59-X%*4: GOSUB 1000*X%
200 rest of program
```

Sure would be nice!

Before I finally say goodbye here are a couple more tips for shortening and speeding-up :

1. Very often you'll find a subroutine which is called only once. Maybe it was originally called more often, but the author will have cleaned up his program, and forgotten to carry out such a check. All you have to do is to make the code 'in-line', and delete the GOSUB call.

2. Even more often you'll find a program line ending with 'GOSUB xxxx: RETURN', or maybe the RETURN is on the following line. In the first case, the 'GOSUB xxxx: RETURN' can be reduced to 'GOTO xxxx', and in the second the final 'GOSUB xxxx' can be replaced by 'GOTO xxxx'. As far as the next line's 'RETURN' is concerned, it may safely be deleted *only if there is no other call to this line.*

SOLUTION TO ABOVE PROGRAM REDUCTION PROBLEM

Hope you didn't peek before having a really good go at it, but here's the solution to our "little" problem above :

```
100 Y% = 18 + (X% + 1) * (X% + 2) / 2
200 rest of program
```

How did I arrive at that? You may very well ask. Let's just say I enjoy mathematical problems of this kind. But, to get you pointed in the right direction, I arrived at a base of 18 by concluding that if X% *could* have taken on values below the '1' in Line 100, the program would surely have looked like this :

```
80 IF X% = -1 THEN Y% = 18
90 IF X% = 0 THEN Y% = 19
```

thus preserving the pattern of increments. That is, first an increment of 1 to move from 18 to 19, then an increment of 2 to go to 21, the first line of our actual program. After that, the rest wasn't too difficult - - comparatively speaking, that is!

First, I subtracted 18 from all values of Y%, including those in my dummy lines, and set the result out in a row, so :

```
N    0   1   2   3    4    5    6    7    8    9 ...

Y%   0   1   3   6   10   15   21   28   36   45 ...
```

where 'N' is Y%'s position in the series. I immediately recognised this series for Y% as being an arithmetical progression with Y% being the sum of any 'N' terms. That is, for example, where N=4 the '10' immediately below it is the sum of all Ns up to that point. The formula for computing such a sum is :

(First N + Current N) * (Number of terms) / 2

In this case, 'first N' is 0, and can be ignored, and 'Number of terms' at any point is one more than N at that point. That is, '4' is actually the 5th term in the series of Ns, and so on. So our formula now becomes :

N * (N + 1) / 2

Check it out and verify it.

Now, below this series I put the corresponding values for X%, commencing at '-1' below the '0's, thus :

```
N    0   1   2   3   4   5   6   7   8   9 ...

X%  -1   0   1   2   3   4   5   6   7   8 ...
```

and immediately noticed that 'N' is equal to 'X% + 1', which I substituted in the formula immediately above to give :

(X% + 1) * (X% + 2) / 2

Of course, I also had to add in the 18 which I'd earlier removed, and that, believe it or not, is how I got my final formula.

To be Continued Next Month

MICROWARE SYSTEMS CORPORATION

NEWS RELEASE

FOR MORE INFORMATION CONTACT:
Andrew Crane, Sales Manager
Microware Systems Corporation
1900 N.W. 114th Street
Des Moines, Iowa 50322
515-224-1929

Des Moines, Iowa -- Microware Systems Corporation announces the release of a new OS-9/68020 C Language compiler optimized for the powerful and robust 32-bit Motorola 68020 microprocessor. The compiler is based on the popular Kernighan & Ritchie standard and includes numerous extensions for the real-time OS-9 Operating System environment.

The new C compiler utilizes the MC68881 math co-processor for high-speed execution of complex math functions. The C compiler also takes full advantage of the MC68020 microprocessor's 32-bit arithmetic instructions and special addressing modes. All compiler/assembler/linker options are controlled by an intelligent compiler executive. This executive relieves the programmer from the necessity of memorizing numerous compiler options and module-calling sequences.

The OS-9/68020 C language compiler includes complete support for the MC68881 math co-processor. The new C language compiler can generate in-line floating-point instructions, link to MC68881 math libraries or trap to a shared, system-wide MC68881 math package. Use of a shared trap module results in programs which load faster and require less memory than programs which have redundant copies of the standard library linked to every program.

The OS-9/68020 C compiler is characterized by a flexible design strategy that allows it to be hosted on any OS-9/680x0-based development system. Both MC68000 and MC68020 code generation packages are included, and either package can be selected using a compile-time option. This strategy permits programmers to effectively develop applications for any 680x0 target system. Completing this total package are the OS-9/68000 and OS-9/68020 optimized assemblers. These assemblers are normally called by the intelligent executive, but they also can be directly used for assembly language programming.

Among the many features of this new C language compiler are numerous extensions for the OS-9 Operating System environment. These extensions include library functions for memory management and system events. The system event functions allow programmers to coordinate different tasks in a real-time application. In addition, a number of library functions provide compatibility with the proposed ANSI standard.

In addition to complete floating-point co-processor support and optimization for the MC68020 microprocessor, the 68020 compiler's library package allows programmers to use C language for writing functions normally requiring assembly language support.

Microware expects its OS-9/68020 C compiler to set a new industry-standard for high-level languages. The OS-9/68020 C compiler is available through Microware's OEM and authorized distributor network, and directly from Microware.

The OS-9 Operating System is a real-time, multi-user and multi-tasking system for computers based on the Motorola family of 68xxx processors. OS-9 is compact, ROMable and provides a UNIX-style environment for application software. Since its introduction in 1983, OS-9/68000 has been licensed to over 350 manufacturers world-wide for use in a variety of industrial, scientific and consumer products.

Founded in 1977, Microware System Corporation specializes in the development of advanced operating systems and programming languages. Last year Sony and Philips announced the OS-9 Operating System as the basis for Compact Disc-Interactive (CD-I) New Media technology. Microware offices are located in Des Moines, Iowa, Santa Clara, California and Tokyo, Japan with field representatives worldwide.

ELECTRONIC MAIL NOW AVAILABLE

Des Moines, Iowa -- Microware Systems Corporation, an internationally recognized pioneer in microcomputer system software development, announces the availability of an Electronic Mail communications program for multi-node networks and development systems.

Mail is a screen- or line-oriented utility specifically designed for Microware's popular OS-9/68000 Operating System. Mail uses your favorite OS-9 editor (uMACS, SCRED, etc.) to create or modify messages. Mail features distributed mailing lists (entire groups accessed at one time) or consecutive mailing lists (from terminal to terminal). Mailing lists can also be defined in terms of other lists.

Received mail can be sent directly to a printer device for immediate printout, spooled on multiuser systems or saved to a file. Mail features on-line help and complete, easy-to-understand documentation.

OS-9, Microware's flagship product, is a real-time, multi-user and multi-tasking operating system for computers based on the Motorola family of 68xxx processors. OS-9 is compact, ROMable and provides a UNIX-style environment for application software. Since its introduction in 1983, OS-9/68000 has been licensed to over 350 manufacturers world-wide for use in a variety of industrial, scientific and consumer products.

Founded in 1977, Microware System Corporation specializes in the development of advanced operating systems and programming languages. Last year Sony and Philips announced the OS-9 Operating System as the basis for Compact Disc-Interactive (CD-I) New Media technology. Microware offices are located in Des Moines, Iowa and Tokyo, Japan with field representatives worldwide.

MOTOROLA INC.

Microprocessor Products Group
6501 William Cannon Drive West
Austin, Texas 78735-8598

EDITORIAL CONTACT:
Mark Vercruysse
512/928-6804

READER CONTACT:
Laura Tolpen
512/440-2035

INQUIRY RESPONSE:
Technical Info Center
P.O. Box 52073
Phoenix, AZ 85072

MOTOROLA DISCONTINUES THE MC3870 AND MC6805T2 MICROCOMPUTERS

Austin, Texas, July 13, 1987... Motorola's Microprocessor Products Group is discontinuing the manufacture of all packaged versions of the following parts: MC3870 and MC6805T2. The devices will be available for lifetime buys for a period of six (6) months following this announcement date with delivery scheduled no more than eighteen (18) months from this announcement date.

The . 3870 is a single-chip 8-bit microcomputer utilizing N-channel technology. The part is vailable for this lifetime buy in a 40-pin Cerdip and a 40-pin plastic DIP.

The MC. 05T2 Microcomputer Unit (MCU) with PLL logic is a member of the M6805 Family of ngle-chip microcomputers. For this lifetime buy the MC6805T2 is only available in 28-pin plastic DIP.

For more infor ation contact your local Motorola Sales Office or authorized Motorola Distributor.

MOTOROLA INC.

Semiconductor Products Sector
P.O. Box 52073
Phoenix, AZ 85072

EDITORIAL CONTACT:
Angela Harfield
(602) 952-3613

READER CONTACT:
John Carey
(602) 821-4426

INQUIRY RESPONSE:
Technical Info. Center
P.O. Box 52073
Phoenix, AZ 85072

MOTOROLA INTRODUCES HIGH PERFORMANCE ECL ARRAY WITH 1K RAM

Phoenix, Arizona, June 30, 1987 . . . Motorola announces the MCA1500M, third in a series of ultra high-performance ECL bipolar arrays built with Motorola's high-density, oxide-isolated MOSAIC II™ process. The MCA1500M array contains logic power of over 1500 equivalent 300 pico-second gates plus 1152 bits at 3.5 ns configurable RAM organized in four blocks of 32 x 9. Predefined memory configurations allow either single port or dual port operation (see table below). The routing flexibility and macrocell structures are designed for next generation high technology system applications.

RAM CONFIGURATION					
Single Address Port			Dual Address Port		
Organization	R Cells	M Cells	Organization	R Cells	M Cells
32 x 9	1	—	32 x 9	2	—
32 x 18	2	—	32 x 18	4	—
32 x 27	3	—	64 x 4	2	—
32 x 36	4	—	64 x 9	4	0.5
64 x 4	1	0.25	128 x 4	4	1.0
64 x 9	2	0.25			
64 x 18	4	0.5			
128 x 4	2	0.5			
128 x 9	4	2.75			
256 x 4	4	1.75			

Notes:
1. R Cells are the number of required 32 x 9 RAM Cells.
2. M Cells are the number of required M Macrocells.

A special design feature of the MCA1500M is dedicated on-chip test circuitry which provides circuit designers guaranteed RAM quality independent of configuration or user-provided test vectors. The array also features two write strobe generators to simplify critical memory timing and to improve performance.

The MCA1500M circuit development cycle time (from design release to shipment of fully tested prototypes) is seven weeks. The typical non-recurring engineering charge is $30K for multiple designs or $37K for a single design. Available in a 149-pin grid array package, the MCA1500M is $195 in 1K quantities and $240 in 100 quantities. Pricing is in U.S. dollars for U.S. delivery only.

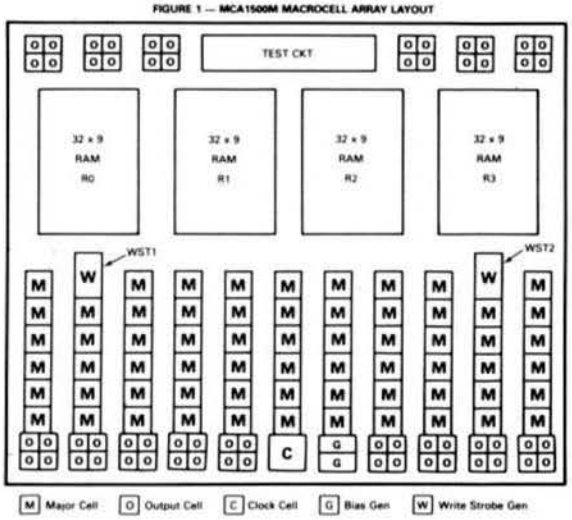
For more information, contact your local Motorola Sales Office or Authorized Motorola Distributor.

• • •

MOSAIC II is a trademark of Motorola Inc.

MCA1500M/D

FIGURE 1 — MCA1500M MACROCELL ARRAY LAYOUT



BASIC MCA1500M ARRAY FEATURES

- Interfaces with MECL 10K/10KH or ECL 100K Logic Families
- Logic Function Specified by User
- Metal Mask Programmable (three unique masks)
- Up to 1700 Equivalent Logic Gates Plus Four Blocks of 32 x 9 (1152 Bits) User Configurable RAM
- On-chip RAM Test Circuitry
- Internal Gate Delays — 0.30 ns typ
- Output Gate Delays — 0.75 ns typ
- 32 x 9 RAM Delays — t_{AA} = 3.5 ns typ
- Supported by Complete CAD Development System
- Power Dissipation — 8.0 Watts typ
- Two Self-Timed Write Strobe Generators
- 149-pin Grid Array Package
- 131 Total Cells with 120 Input/Output Ports
- Major Cell Delays — 0.25 to 1.25 ns typ (0.35 to 1.77 max) depending on logic function
- Output Cell Delays — 0.5 to 1.0 ns typ (0.65 to 1.5 ns max) depending on logic function
- RAM Address Access Time — 3.5 ns typ (5.0 max)
- High-impedance Pulldown Resistors on All Input and Bidirectional Pins
- Up to 64 Total Outputs of which up to 55 Can Drive 25-ohm Loads (all outputs can drive 50-ohm loads)
- Output Edge Speed — 1.0 ns typ 20 to 80% (0.4 ns min 20 to 80% or 0.6 ns min with edge rate slow down option)
- Ambient Temperature Range (with heat sink and 750 lfpm air) — 0 to 70°C
- Temperature Coefficient is 3.3°C/W typ with Heat Sink and 750 lfpm air flow
- Voltage Compensated — V_{EE} = −4.2 to −5.46 Volts

MOTOROLA INC.

Semiconductor Products Sector
P.O. Box 52073
Phoenix, AZ 85072

EDITORIAL CONTACT:
Angela Hathaway
(602) 952-3613

READER CONTACT:
John Carey
(602) 821-4420

INQUIRY RESPONSE:
Technical Information Center
P.O. Box 52073
Phoenix, AZ 85072

12 X 12-BIT ECL MULTIPLIER CIRCUITS SPEED UP ARITHMETIC OPERATIONS

Phoenix, Arizona, May 20, 1987 . . . Motorola has announced the MC10951 series of 12 x 12-bit expandable high-speed multiplier circuits intended for real-time signal processing applications where processing speed is of primary importance. Made with subnanosecond ECL technology, the monolithic LSI circuits perform a typical multiplication function, P = X times Y + M + K, in about 12 nanoseconds.

Several types of multiplication modes are available:

- Multiplying two 12-bit 2's complement numbers produces a 24-bit 2's complement output.
- Multiplying two 12-bit magnitude-only numbers produces a 24-bit magnitude-only product.
- Multiplying a 12-bit magnitude-only number by a 12-bit 2's complement number produces a 24-bit 2's complement output.
- Multiplying two 13-bit signed magnitude numbers (12-bit magnitude, 1-bit sign) produces a 25-bit signed magnitude product.

The MOSAIC II technology based parallel multiplier features a number of unique and important attributes. Specifically:

- The circuits can be used in arrays to perform larger multiplications. Any number of bits can be multiplied by using multiple parts in an array configuration. The number of parts required for expanded inputs, together with the resulting multiplication time is given in the following table.

Size of Operands	Number of Parts	Multiplication Time (ns) Typ	Max
12 x 12	1	12	21.4
24 x 24	4	22.4	38.6
48 x 48	16	42	73.6

- The circuit performs IEEE standard floating point operations with shift bit correction.
- It can perform in a pipeline configuration which allows N x N multiplication in the time needed for a single 12 x 12-bit operation.
- It has two 24-bit outputs — latched for product outputs and unlatched for fast partial-product summation in the array.
- The circuits are specified around the dc specification of the MECL 10KH family but are fully compatible with all other MECL families. A special ECL 100K compatible option is also available.

The three available options, all packaged in a 149-pin grid array package, and pricing for the MC10951 series of multiplier circuits is listed below in U.S. dollars for U.S. delivery only. Delivery is stock to 60 days.

Device	Description	Price	Quantity
MC10951R	10K/10KH compatible at V_{EE} = −5.2 Vdc ± 5%	$250.00	100
MC10L951R	10K/10KH compatible at V_{EE} = −4.5 Vdc ± 0.3 Vdc	250.00	100
MC100951R	100K compatible at V_{EE} = −4.5 V ± 0.3 Vdc	250.00	100

For further information, contact your local Motorola Sales Office or Authorized Motorola Distributor.

###

DOMINIC EVANS INDUSTRIES

COMPLETE 68000 COMPUTER MOUNTS ON A 5.25 INCH DISK DRIVE

Dominic Evans Industries announces immediate availability of the DE168K stacking two-board computer system.

The system is aimed at users developing very high performance systems in applications such as Industrial Control, Data Logging and High Resolution Graphics Systems.

Based on a 12MHz 68000 or 68010 32-Bit Microprocessor, and not requiring an expensive backplane, the system offers high performance at low cost.

The DE168K is supplied with 512KByte of on-board ZERO-WAIT-STATE CMOS STATIC RAM, fitted into Battery-Backed 32-pin JEDEC memory sites, which will allow 2.0MBytes of RAM to be fitted when 1MBit devices become available.

Interfaces include a Fully Arbitrating Initiator/Target SCSI Controller which can support multiple Hard Disk Drives, Tape Streamers, CD-ROMs etc., and a Flexible Disc Controller supporting upto four 1MByte Drives, together with four RS232 Serial ports, or three RS232 Serial ports and one RS485 Serial port for NETWORKING at data rates of 250KBits/Sec.

Other features include a Battery-Backed Real-Time Clock/Calendar, four Parallel ports with a Centronics interface and Bus Buffering/Decode to support additional stacking boards, which include a Very High Performance Graphics System with Maths Co-processor.

DEI-BUG68K, an extensive ROM-Based DEBUG MONITOR is which is available, provides facilities to Upload/Download Motorola S-Format Binary files between Host and Target systems, together with Disk Operating Bootstraps and Diagnostics.

A fully configured operating system with Native Assembler, C-Language Compiler and Screen Editor provides a complete development environment for 68000 systems.

A unique 6809-FLEX Simulator allows users to run their existing library of 8-Bit FLEX software, and to develop new 6809 software in an enhanced environment.

The MDS68K 68XXX Development Work Station incorporates the DE168K Two-Board Computer System together with one or two 1MByte Flexible Disk Drives and an optional 25.5MByte Hard Disk Drive. The system is fully enclosed in a robust steel enclosure with a footprint slightly larger than a 5.25 inch disk drive.

COMPLETE 6809 BASED COMPUTER SYSTEM ON A SINGLE EUROCARD

Dominic Evans Industries announces immediate availablity of the DE109 Single Board Computer and the MDS09 Microprocessor Development System.

The systems are aimed at users developing industrial control systems or users requiring a general purpose high performance computer for scientific and research applications.

Based on a 2MHz 6809 microprocessor, the Single Board Computer features upto 8KBytes of EPROM, two 28 pin RAM/ROM sockets with 64KBytes of NON-VOLATILE RAM, a Flexible Disk Controller supporting upto four 1MByte drives, two RS232 Serial ports with independently programmable baud rates upto 38400Baud, two Parallel ports with two 16-bit Counter/Timers, Centronics interface and connectors for a multi-line LCD module and parallel keyboard.

Other features also offered as standard, are a Battery-Backed Real-Time Clock/Calendar, and a System watchdog Timer.

The board operates from a SINGLE unregulated DC supply, using on-board regulation for system power, and a DC-DC converter for reliable +/- 12Volt RS232 operation.

To provide the highest reliability, all devices are mounted in Gold Turned-Pin sockets.

System expansion is provided for with a Fully Buffered Bus Connector allowing additional cards to be connected for larger systems. Expansion cards can be fitted to provide upto 1MByte of NON-VOLATILE RAM, Colour Graphics Systems Digital/Analogue I/O and Quadrature Encoder Interfaces.

The Single Board Computer is available in several configurations starting from £250-00, allowing the OEM user to choose the price/performance level required.

Also available is DEI-BUG09, a powerful EPROM-Based DEBUG MONITOR which includes Diagnostics, Disk Operating Bootstraps and facilities to Upload/Download Motorola S-Format Binary files between Host and Target systems.

The MDS09 Micro Development System uses the Single Board Computer and is a fully packaged stand alone system, with two 5.25 inch flexible disk drives, and enhanced FLEX development software.

For further information please contact:

Roger Shingler at DOMINIC EVANS INDUSTRIES on 0922-57853.

68 Micro Journal - 10th Year Reader Survey

As has been our practice in the past, we are again seeking your answers to a few questions. This time we are attempting to make it simpler. All you have to do is fill out and remove this page, fold in half, staple or tape, and drop in the mail box. You won't damage any regular material as both sides are related to this survey. We would appreciate your signing the form, but that isn't required. We are more interested in what you actually think.

After 10 years we would like to know what you think about our performance. Also we want to know what you think about the performance of any of our present or past advertisers you might have had dealings with. Your replies will give us direction on what we might consider in our planning for the future.

As to your replies, we would like to know the **"Good - Bad & the Ugly"**. If we don't have your input we can't fulfill our pledges made to you over the years. Please take a little time and let us know.

Again what we want to find out is how - **"ALL OF US - PAST & PRESENT"** - have delivered as we promised, not only now, but for the entire time you have been a subscriber, or when you first purchased anything from any of our present or past advertisers.

Please complete and return within 30 days - thanks

What I like or dislike about 68 Micro Journal: ______________________________

____

____

____

For the items below, article could be column, use additional paper if necessary

My favorite author: ____________ Article or Column: ____________ Date ______

The best article ever: ____________ Why: ____________ Date ______

The best advertiser I had dealings with: ____________ Item ______

Why I rate them best: ______________________________

____

The worse advertiser I had dealings with: ____________ Item ______

Why I rate them worse: ______________________________

____

Any additional remarks: ______________________________

____

Name: ______________________________ Date ______

Please tear-out, fold along this line and staple

From: ____________________

____

____

____

Place Stamp Here

68 Micro Journal
Computer Publishing Center
POB 849
Hixson, TN 37343

Staple or tape here

A PL/9 interface for ISAM

by Martin C. Gregorie
10 Sadlers Mead
Harlow, Essex U.K.

Continued From Last Month

```
        +10        Pointer to record buffer
        +11        record length (copy of +24)

        +12        Last reply code returned by ISAM

        +13        ISAM file number as a 16 bit unsigned number

        +14-+21    ISAM file name string. Up to 14 bytes plus final null

        +22        Key length in bytes as a 16 bit unsigned number (this is
                   used by obtain to see if the record requested is the one
                   returned).

        +23        File size in records as a 16 bit unsigned number

        +24        Record size in bytes as a 16 bit unsigned number

/* LABEL - ISAM test program

        written by Martin Gregorie, January 1987
                   10 Sadlers Mead
                   Harlow, Essex, U.K.
*/

origin=$2000;
stack=*;

include minio;              /* see DISNAME article */
include environ;            /* see DISNAME article */
include isam;

byte help "\n\n
  LABEL program commands\n
----------------------------\n
OPEN    CLOSE    CREATE\n
FIRST   LAST     START\n
GET     CHANGE\n
NEXT    CURRENT  PREVIOUS\n
ADD     DELETE   REORGANISE\n
END     LIST     INITIALISE\n
HELP\n
----------------------------\n";

procedure readkey(byte .key);
   print("\nKey : ");
   input(.key,30);
endproc;

procedure clearrec(byte .name,.street,.city,.zip);
   name=nul; street=nul; city=nul; zip=nul;
endproc;

procedure inputrec(byte .name,.street,.city,.zip);
   print("\nName       "); input(.name,30);
   print("\nStreet     "); input(.street,30);
   print("\nCity       "); input(.city,30);
   print("     Zip "); input(.zip,10); crlf;
endproc;

procedure showrec(byte .name,.street,.city,.zip);
   print("\nName       "); print(.name);
   print("\nStreet     "); print(.street);
   print("\nCity       "); print(.city);
   print("     Zip "); print(.zip); crlf;
endproc;

procedure list_records(integer .icb:
                       byte .name,.street,.city,.zip);
   obtain(.icb,.first,.name);
   while dbstatus(.icb)=0
      begin
         showrec(.name,.street,.city,.zip);
         obtain(.icb,.next,.name);
      end;
endproc;

byte cmnds "HE","OP","CL","CR","FI","LA","ST","GE","CH",
           "NE","CU","PR","AD","DE","RE","IN","LI",
           "EN","XX";

procedure label;
   byte icb(30),
        file(16),command(16),key(31);
   integer size, recsize,cno,i;
   byte name(31),street(31),city(31),zip(11);
   print(.help);
   clearrec(.name,.street,.city,.zip);
   repeat
      print("\n\nEnter command ");
         input(.command,13);
         command(2)=endstr;
         toupper(.command);
         i=0;
         cno=0;
         repeat
            if equal(.command,.cmnds(i))
               .or equal("XX",.cmnds(i)) then break;
            i=i+3;
            cno=cno+1;
         forever;
         if cno
            case 0 then print(.help);
            case 1 then      /* OPEN */
               begin
                  print("\nEnter file name "); input(.file,15); crlf;
                  ready(.icb,0,.file,104,30);
               end;
            case 2 then finish(.icb);    /* CLOSE */
            case 3 then      /* CREATE */
               begin
                  print("\nEnter file name "); input(.file,15);
                  print("\nNo of records   "); size=inputint; crlf;
                  format(.icb,0,.file,size,104,30);
               end;
            case 4 then position(.icb,.first);      /* FIRST */
            case 5 then position(.icb,.last);       /* LAST */
            case 6 then      /* START */
               begin
                  readkey(.key);
                  position(.icb,.key);
               end;
            case 7 then      /* GET */
               begin
                  readkey(.key);
                  obtain(.icb,.key,.name);
                  showrec(.name,.street,.city,.zip);
               end;
            case 8 then      /* CHANGE */
               begin
                  inputrec(.name,.street,.city,.zip);
                  modify(.icb,.name);
               end;
            case 9 then      /* NEXT */
               begin
                  obtain(.icb,.next,.name);
                  showrec(.name,.street,.city,.zip);
               end;
            case 10 then      /* CURRENT */
               begin
                  obtain(.icb,.current,.name);
                  showrec(.name,.street,.city,.zip);
               end;
            case 11 then      /* PREVIOUS */
               begin
                  obtain(.icb,.previous,.name);
                  showrec(.name,.street,.city,.zip);
               end;
            case 12 then      /* ADD */
               begin
                  inputrec(.name,.street,.city,.zip);
                  store(.icb,.name);
               end;
            case 13 then erase(.icb);       /* DELETE */
            case 14 then reorganise(.icb);       /* REORGANISE */
            case 15 then abort(.icb);       /* INITIALISE */
            case 16 then list_records(.icb,.name,.street,.city,.zip);
            case 17 then print("\n          Run ended");
            else
               print("\n          What??");
         if dbstatus(.icb) .and cno .and cno<16 then
            begin
               print("\nDB error "); printint(dbstatus(.icb)); crlf;
            end;
      until equal(.command,"EN");
```

```
 NAM ISAMSIM
*
* ISAM.CMD INTERFACE TESTER
*
*  WRITTEN BY MARTIN GREGORIE, JANUARY 1987
*            10 SADLERS MEAD
*            HARLOW, ESSEX, U.K.
*
* EQUATES
*
* TEST CODE BASE
*
BASE EQU $B800
RAMTOP EQU BASE-1
PPTR EQU RAMTOP-4
*
```

```
* Flex entry points
*
WARMS EQU $CD03 FLEX WARM START
PUTCHR EQU $CD18 PUT CHARACTER
PCRLF EQU $CD24 CR/LF
OUTDEC EQU $CD39 O/P DECIMAL NUMBER
OUTHEX EQU $CD3C O/P HEX BYTE
OUTADR EQU $CD45 O/P 4 HEX DIGITS
*
* TEST DATA STRUCTURE
*
 ORG $BE00
*
STRUCT FDB SFILENO,2
 FDB SCOMD,2
 FDB SFILSP,14
 FDB SREC,2
 FDB SFLEN,2
 FDB SBUFF
SREC FDB 50
SERR FDB 2
SFILENO FDB 1
SFILSP FCC '1.TESTFILE.ISA',0,0
SKLEN FDB 17
SFLEN FDB 1000
 FDB 0
SCOMD FCC 'CO',0
SBUFF FCC 'RECORD BUF'
 FCC 'FER HOLDIN'
 FCC 'G UP TO 50'
 FCC 'BYTES RADI'
 FCC 'NG HERE ->',0
**
*
* MEMEND RESET
*
 ORG $CC2B
 FDB RAMTOP
*
* START OF TESTING CODE
*
 ORG BASE
*
*
*
LISTER LDX #MSG1 PRINT HEADER LINES
 JSR PSTRNG
 LDY PPTR GET PARAM POINTER
 LEAU 0,Y
 LDD #0 PARAM 0
 JSR HDDR
 LDX 0,U
 BEQ L1 NO PARAM
 CLR B
 JSR OUTDEC FILE NUMBER
L1 LDY PPTR
 LEAU 4,Y
 LDD #1 PARAM 1
 JSR HDDR
 LDX 0,U COMMAND
 BEQ L2
 JSR PSTRNG
L2 LDY PPTR
 LEAU 8,Y
 LDD #2 PARAM 2
 JSR HDDR
 LDX 0,U FILE NAME
 BEQ L3
 JSR PSTRNG
L3 LDY PPTR
 LEAU 12,Y PARAMETER 3
 LDD #3
 JSR HDDR
 LDX 0,U RECORD SIZE
 BEQ L4
 CLR B
 JSR OUTDEC
L4 LDY PPTR
 LEAU 16,Y PARAMETER 4
 LDD #4
 JSR HDDR
 LDX 0,U FILE SIZE
 BEQ L5
 CLR B
 JSR OUTDEC
L5 LDY PPTR
 LEAU 20,Y PARAMETER 5
 LDD #5
 JSR HDDR
 LDX 0,U RECORD BUFFER
 BEQ L6
 JSR PSTRNG
L6 LDX #MSG5
 JSR PSTRNG
 LDD #123 ERROR CODE
 STD PPTR RETURNED
 RTS
*
* PARAM POINTER DETAILS
*
HDDR LDX #MSG2
 JSR PSTRNG
 PSHS A,B
 LEAX 0,S
 CLR B
 JSR OUTDEC PARAMETER NUMBER
 PULS A,B
 LDX #MSG3
 JSR PSTRNG
 LEAX 0,U
 JSR OUTADR POINTER TO PARAMETER
 LDA #'/
 JSR PUTCHR
 LEAX 2,U
 CLR B
 JSR OUTDEC LENGTH OF PARAMETER
 LDX #MSG4
 JSR PSTRNG
 RTS
*
*
*
PSTRNG LDA 0,X+
 BEQ PEX
 JSR PUTCHR
 BRA PSTRNG
PEX RTS
**
*
* MESSAGES
*
MSG1 FCB $0A,$0D
 FCC 'ISAMSIM Parameter ar   dump'
 FCB $0A,$0D
 FCC '---------------------------',0
MSG2 FCB $0A,$0D
 FCC 'Parameter ',0
MSG3 FCC ' at ',0
MSG4 FCC ' :',0
MSG5 FCB $0A,$0D
 FCC '---------------------------'
 FCB $0A,$0D,0
*
* TESTER TESTING ENTRY CODE
*
TT LDX #STRUCT
 STX PPTR
 JSR LISTER
 JMP WARMS BACK TO FLEX

 END TT

/* ISAM.LIB - ISAM.CMD interface procedures - V1.0

             written by Martin Gregorie, January 1987
                        10 Sadlers Mead
                        Harlow, Essex, U.K.

   All verbs use an Isam Control Block which should be declared as
   a 50 byte area.

*/

at $CC2B: integer memend;

/*
   The following name common ISAM fail codes as constants
*/

constant db_status_ok=0,
         db_start_of_file=165,
         db_end_of_file=170,
         db_file_full=180,
         db_rec_not_found=-100;
/*
   The following are not real at-variables but special commands
   recognised by their high negative addresses. True keys must be
   below $Ca00!
*/

at $fff0: byte first(3),last(3),next(3),current(3),previous;

byte isrdy "OP";                    /* ISAM command strings */
byte isfin "CL";
byte isabt "IN";
byte ispos "FI","LA","NE","CU","PR";
byte isrct "ST";
byte isadd "AD";
byte isdel "DE";
byte isrep "RE";
byte iscrt "CR";

/*
   Support procedures
   ------------------
*/

/* ISAM call interface */

asmproc isam(integer);
   GEN $34,$20;        /*         PSHS   Y        */
   GEN $BE,$CC,$2B;    /*         LDX    MEMEND   */
   GEN $EC,$64;        /*         LDD    4,S      */
   GEN $ED,$1C;        /*         STD    -4,X     */
   GEN $AD,$01;        /*         JSR    1,X      */
   GEN $BE,$CC,$2B;    /*         LDX    MEMEND   */
   GEN $EC,$1C;        /*         LDD    -4,X     */
   GEN $AE,$64;        /*         LDX    4,S      */
   GEN $ED,$88,$18;    /*         STD    24,X     */
   GEN $35,$20;        /*         PULS   Y        */
   GEN $39;            /*         RTS             */
```

To be Continued Next Month

Classifieds
As submitted - No Guarantees

DAISY WHEEL PRINTERS
Qume Sprint 9 - $900 Qume Sprint 5 - $800

HARD DISK 10 Megabyte Drive - Seagate Model #412 $275.

3 - Dual 8" drive enclosure with power supply . New in box . $125 each.

5 - Siemens 8" Disk Drive , $100 each.

Tano Outpost II, 56K, 2 5" DSDD Drives, FLEX, MUMPS, $495.

TELETYPE Model 43 PRINTER - with serial (RS232) interface and full ASCII keyboard . $250 ready to run.

SWTPC S/09 with Motorola 128K RAM, 1-MPS2, 1-Paralell Port, MP-09CPU Card - $900 complete.

(615) 842-4600 M-F 9AM to 5PM EST

SWTPC 6809 system 5709 , dual 8" drives, 8212 terminal, 128K RAM, Okidata wide carriage with tractor feed . $2,000 or best offer.
Pete Yore, P.O. Box 384, Largo, Fl 34294-0384
(813)462-0511 or (813)536-0018

•••

Have many 6800 and 6809 bare and working boards and manuals . Send SASE for list.
Ron Mauceri, 2037 Dewberry Ct.,Westlake Village, CA 91361

•••

GMX 68020 Computer, Never used.
16.7 MHz. 2 Meg Ram. floppy. 25 Meg HD.
4 serial. 1 par. With C Uniflex OS. et. al.
$5500 new. Only $3995. Mark Talisman.
(714) 582-9100 or (714) 532-3466 eves.

!!! Subscribe Now !!!
68 MICRO JOURNAL

Subscription Rates

U.S.A.: 1 Year $24.50, 2 Years $42.50, 3 Years $64.50

•Foreign Surface: Add $12.00 per Year to USA Price.

•Foreign Airmail: Add $48.00 per Year to USA Price.

•Canada & Mexico: Add $9.50 per Year to USA Price.

•U.S. Currency Cash or Check Drawn on a USA Bank !

68 Micro Journal
5900 Cassandra Smith Rd.
POB 849
Hixson, TN 37343

Telephone 615 842-4600

Telex 510 600-6630

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Mastercard ☐ VISA ☐

Card #______________________ Exp. Date__

For 1 Year ___ 2 Years ___ 3 Years __

Enclosed: $__________

Name ______________________

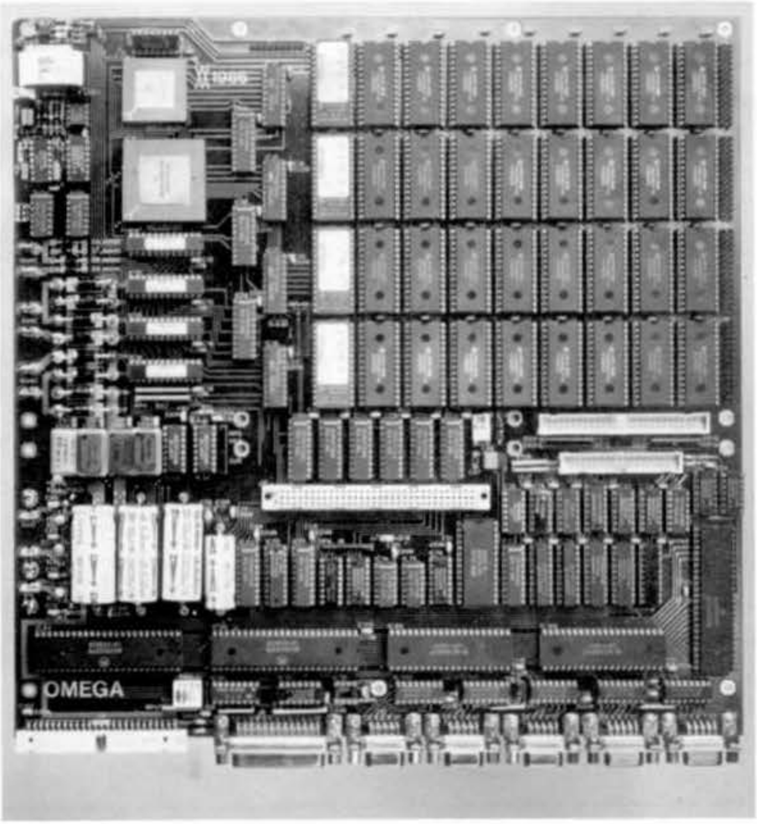
Street ______________________

City ____________ State ____ Zip ______

My Computer Is: ____________

FOR THOSE WHO NEED TO KNOW

68 MICRO JOURNAL™



12" x 12"

On-board power supply . . . only requires external 10 VA transformer.

5 x RS-232C serial ports.

SCSI initiator.

FDC controller

Parallel printer port.

16-bit bi-directional I/F

Non-volatile clock calendar

Fullybuffered bus expansion

QUANTITY DISCOUNTS
2-5 less 5%, 6-9 less 10%
10-24 less 15%, 25-99 less 20%
100+ less 25%

12.5 MHz MC68020 32-bit processor and 12.5 MHz MC68881 Floating Point Coprocessor plus 1 Mb non-volatile, zero wait-state 100% STATIC RAM all AS STANDARD!

Ωmega . . . The OEM's CHOICE

OMEGA/12.5	OMEGA w/12.5 MHz 020/881 & 1 MB RAM	$2750.00
OMEGA/16.6	OMEGA w/16.67 MHz 020/881 & 1 MB RAM	$3295.00
OMEGA/LTX	OMEGA 120/240 VAC Line Transformer	$ 22.00
OMEGA/1MB	1 megabyte additional STATIC RAM	$ 895.00
OMEGA/SER9	9 port RS-232C serial expansion board	$ 645.00
OMEGA/GRF	640 x 480 x 4 bits/pixel ACRTC graphics I/F	$ 925.00
OMEGA/OS9	OS-9/68K™ 'Professional' configured for OMEGA	$ 595.00

PRICES INDICATED INCLUDE SHIPPING AND APPLY TO U.S. CUSTOMERS ONLY

VISA/MASTER CARD ORDERS ACCEPTED

OS-9 is a trademark of Microware Systems Corporation

NON-U.S. CUSTOMERS SHOULD CONTACT THE NEAREST DEALER FOR PRICE & DELIVERY INFORMATION

NORTH & SOUTH AMERICA

LLOYD I/O INCORPORATED
19535 NE GLISAN
P.O. Box 30945
PORTLAND, OR 97230 (USA)
TEL: (503) 666-1097
TLX: 9103805448 LLOYD I O

ALL OTHER ENQUIRIES

WINDRUSH MICRO SYSTEMS
WORSTEAD LABS.
N. WALSHAM, NORFOLK
NR28 9SA, ENGLAND
TEL: (0692) 404086
TLX: 975548 WMICRO-G

CONTINENTAL EUROPE

SNIJDER MICRO SYSTEMS
SCHOOTEINDSEWEG 8a
5756 BD Vlierden
The NETHERLANDS

TEL: (0)4930-11975-13666

LLOYD I/O

TM INC.

ESTABLISHED 1979

ASSEMBLY LANGUAGE DEVELOPMENT SYSTEMS

VANTAGE™ [illegible]

The [illegible] talk to each other directly with out having to exchange data on slow disk files. In fact the only [illegible]

VANTAGE [illegible]

The [illegible] referencing, 10 edit buffers, command macros, side file functions, access to OS9, etc. When you are ready to assemble, you simply call the cross assembler. It assembles the source code [illegible]

CRASMB is packed with many features to speed your development job. Powerful macro features, symbol cross referencing by line numbers, and plain English error messages are among these. **CRASMB** offers the best in compatibility with the manufacturers' assembly [illegible]

CRACKER [illegible]

CRACKER [illegible]

[illegible] are the key in testing software, so we went all out in our research and development of the breakpoint system. We looked at other debuggers on several different computers, and talked with many assembly language programmers. **CRACKER** has easy to use breakpoints, but at the same time they are very powerful, including: conditional breakpoints, unlimited scope, memory protection, many types of actions including the execution of a macro-breakpoint-language for simulation of hardware devices. Expressions may use symbols that were defined in the assembly process to specify memory address values. Breakpoints remember the symbol names making it easy to make changes to your source code without having to kill and enter new breakpoints. The simulator allows you to execute your object code in a controlled environment. If things get out-of-wack, your memory protection and breakpoints will keep the simulation under control. Single stepping displays the registers and either a dis-assembly of the instruction about to be executed or a listing of the source code line from the editor that generated the object code, i.e. source code debugging is built in.

SERVICE includes a 1 year free update period from the time of purchase. We ship within 4 working days and use UPS BLUE. Each copy is custom manufactured to your computer disk size and format.

Supported CPU's

CRASMB	CRACKER
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	BY [illegible]
[illegible]	BY [illegible]
[illegible]	BY [illegible]
[illegible] 8741	BY [illegible]
8048,20,21,22 ETC.	BY [illegible]
[illegible]	BY [illegible]
[illegible]	BY [illegible]
Z8	BY 3Q87
Z80	BY 4Q87

For OS9/68000 .... $795

[illegible]

[illegible]	$437
[illegible]	[illegible]
[illegible]	[illegible]
[illegible] (OS9/68000)	$495

DEALER TELEPHONES

USA	
SE S.E. Media Supply	1 615 842 4600
NE Frank Hogg Lab. Inc.	1 315 474 7856
SW GESPAC, Inc.	1 602 962 5559
NE Thomas Instrumentation	1 609 967 4280
NW D & S Industries	1 206 827 7300
NW Stylo Software	1 208 529 3210
AUSTRALIA	
Paris Radio Electronics	0 2 344 9111
DENMARK	
H.C. Andersen Computer A/S	0 1 52 44 04
ENGLAND	
Vivaway Ltd.	0 582 423 425
Windrush Micro Systems	0 692 404 086
FRANCE	
Microdata-Soft	1 768 8080
Computer Dialysis France	1 4789 8442
WEST GERMANY	
Dr. Rudolf Keil GmbH	0 62 03 67 41
Zacher Kleincomputer	0 65 25 299
JAPAN	
Seikou Electronics Co.	03 832 6000
Microboards, Inc.	0 474 22 1741
SWEDEN	
Micromaster Scandinavian AG	018 138595
SWITZERLAND	
Elsoft AG	056 83 33 77

VISA – MASTERCARD

DEALER and OEM inquiries invited

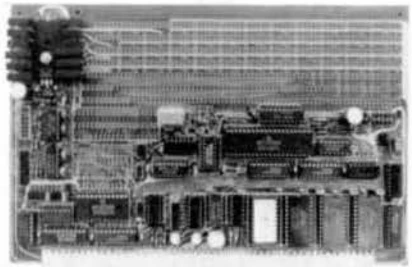
VANTAGE, ED, CRASMB, CRACKER are trademarks of LLOYD I/O, INC.

[illegible]

LLOYD I/O, INC. • P.O. Box 30945 • Portland, OR 97230 • USA
Attn.: Frank Hoffman • Phone 1-503-666-1097 • Telex 910 380 5448 LLOYD IO

E-Z On, E-Z Off

The LAB 6809 helps you prototype your design in less time, for less money. Easy to use and easy to modify, the LAB 6809 is the optimal development tool for 6809-based applications with high performance objectives.



☐ Use as stand-alone or system CPU ☐ 68B09 jumper selectable for 1 or 2 MHz ☐ 2 MHz I/O without wait states for on-board devices ☐ 68B50 RS-232C serial port ☐ on-board provisions for additional 6850, 6840, 6821, other I/O devices ☐ PICOBUG monitor provides I/O and debugging routines ☐ 22 sq. in. user wirewrap area ☐ gold-plated SS-50 connectors ☐ comprehensive documentation

Put the LAB 6809 to work on your next project. $395.

Shipping and handling $10 (US), $20 (outside US). Texas residents add sales tax. MasterCard, VISA accepted. Technical bulletin available on request. Dealer inquiries invited.

Barton Laboratories INTELLIGENT TOOLS FOR INTELLIGENT CONTROLS

1606 West 14th Street Austin, TX 78703 512/477-6809

Installed Systems World-Wide
OVER 10 YEARS OF DEDICATED QUALITY!

CPI *A Division of Computer Publishing, Inc.*
5900 Cassandra Smith Road
Hixson, Tn 37343
Telephone 615 842-4600
Telex 510 600-6630

DATA-COMP SPECIAL
Heavy Duty Power Supplies

For A limited time we are offering our HEAVY DUTY SWITCHING POWER SUPPLY. These are BRAND NEW units and will not last long. Also note that these prices are less than 1/4 the normal price for these high quality unit.

Make: Boschert

Size: 10.5 x 5 x 2.5 inches - including heavy mounting bracket and heatsink.

Rating: in 110/220 volts ac (strap change) Out: 130 watts

Output: +5v - 0 amps
+12v - 4.0 amps
+12v - 2.0 Amps
-12v - 0.5 amps

Mating Connector: Terminal strip
Load Reaction: Automatic short circuit recovery

Each
SPECIAL: $59.95
2 or more 49.95

Add: $7.50 each S/H

Make: Boschert

Size: 10.75 x 6.2 x 2.25 inches

Rating: 110/220 ac (strap change) Out: 81 watts

Outputs: +5v - 8.0 amps
+12v - 2.4 amps
+12v - 2.4 amps
+12v - 2.1 amps
-12v - 0.4 amps

Mating Connectors: Molex
Load Reaction: Automatic short circuit recovery

Each
SPECIAL: $49.95
2 OR MORE 39.95

Add: $7.50 S/H each

SK★DOS™

The Generic DOS™ for 68000 applications in

- ★ Industrial Control
- ★ Business Use
- ★ Educational Computing
- ★ Scientific Computing
- ★ Number Crunching
- ★ Dedicated Systems
- ★ Turnkey Systems
- ★ Data Collection
- ★ Single board Computers
- ★ Bus oriented Computers
- ★ Graphics Workstations
- ★ One-of-a-kind Systems
- ★ Advanced Hobbyist Use

SK★DOS is a single-user disk operating system for computers using Motorola 32 bit CPUs such as the 68008, 68000, 68010, and 68020. It provides the power of a full DOS, yet is simple and easy to use, and will run on systems from 32K to 16 megabytes. Because SK★DOS is easily implemented on a new system, we call it "The Generic DOS" which allows programs written for one system to be run on many others.

SK★DOS comes with over 40 commands and system programs, including a 6809 emulator which allows 68K SK★DOS to run application programs and languages developed for 6809 SK★DOS and other systems. Assemblers, editors, and higher level language support are available from third party software vendors and through public domain software.

SK★DOS is available for single copy or dealer sales, as well as OEM licensing. Single copies cost $125 (inquire as to available systems). Extremely attractive OEM licensing terms are also available. An optional Configuration Kit contains a detailed Configuration Manual and two disks of source code for system adaptation, including source code for a system monitor/debug ROM and other programs useful for adapting SK★DOS to new systems.

SK★DOS *is available from*

STAR-K SOFTWARE SYSTEMS CORPORATION
P.O. BOX 209 · MT. KISCO, NY 10549 · 914/241-0287
TELEX 510601 6774

INDUSTRIAL PASCAL FOR 68000 AND 6809

PCSK is a package that generates code for a 68000 series processor running on a 68000 development system. It includes the compiler, assembler, linker, host debugger, target debugger, and screen editor, all integrated together and controlled by a menu driven shell program. Source code is included for the runtime library and many of the utilities. Host operating systems supported are OS-9/68000 (Microware), PDOS (Eyring Research), and VERSAdos (Motorola).

PXK9 is a package that generates code for a 6809 processor running on a 68000 development system. Includes all of the features of the PCSK package above, except for the host debugger. Host operating system is OS-9/68000.

I WANT IT, WHERE DO I GET IT?

For more information on either of these two products please contact Certified Software, South East Media, or one of our European Licensees.

OEM LICENSEES

Gespac sa. 3, chemin des Aulx, CH-1228 Geneva/Plan-les-Ouates, Switz. TEL (022) 713400, TLX 429989

PEP Elektronik Systeme GmbH, Am Klosterwald 4, D-8950 Kaulbeuren, West Germany. TEL (08341) 8974, TLX 541233

Eltec Elektronik GmbH, Galileo-Galilei-Strasse, 6500 Mainz 42, Postfach 65, West Germany. TEL (06131) 50031, TLX 4187273

DISTRIBUTORS

R.C.S. Microsystems Ltd, 141 Uxbridge Road, Hampton Hill, Middlesex, England. TEL 01-9792204, TLX 8951470

Dr. Rudolf Keil GmbH, Porphystrasse 15, D-6905 Schriesheim, West Germany. TEL 062 03/6741, TLX 465025

Elsoft AG, Zelgweg 12, CH-5405 Baden-Daettwil, Switzerland. TEL 056-833377, TLX 828275

Byte Studio Borken, Butenwall 14, D-4280 Borken, West Germany. TEL 02861-2147, TLX 813343

CERTIFIED SOFTWARE CORPORATION
616 CAMINO CABALLO, NIPOMO, CA 93444
TEL: (805) 929-1395 TELEX: 467013
FAX: (805) 929-1395 (MID-8AM)

SOFTWARE FOR 680x AND MSDOS

SUPER SLEUTH DISASSEMBLERS
EACH $99-FLEX $101-OS/9 $100-UNIFLEX
OBJECT-ONLY versions: EACH $50-FLEX,OS/9,COCO
interactively generate source on disk with labels, include xref, binary editing
specify 6800,1,2,3,5,8,9/6502 version or Z80/8080,5 version
OS/9 version also processes FLEX format object file under OS/9
COCO DOS available in 6800,1,2,3,5,8,9/6502 version (not Z80/8080,5) only
NEW: 68010 disassembler $100-FLEX,OS/9,UNIFLEX,OS/9-68K,MSDOS

CROSS-ASSEMBLERS WITH MACRO CAPABILITIES
EACH $50-FLEX,OS/9,UNIFLEX,MSDOS,UNIX 3/$100 ALL/$200
specify: 180x,6502,6801,6804,6805,6809,Z8,Z80,8048,8051,8085,68010,32000
modular cross-assemblers in C, with load/unload utilities   **NOW: OS/9-68K**
sources for additional $50 each, $100 for 3, $300 for all

DEBUGGING SIMULATORS FOR POPULAR 8-BIT MICROPROCESSORS
EACH $75-FLEX $100-OS/9 $80-UNIFLEX
OBJECT-ONLY versions: EACH $50-COCO FLEX,COCO OS/9
interactively simulate processors, include disassembly formatting, binary editing
specify for 6800/1, (14)6805, 6502, 6809 OS/9, Z80 FLEX

ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809
6502 to 6809 $75-FLEX $85-OS/9 $80-UNIFLEX
6800/1 to 6809 & 6809 to position-ind. $50-FLEX $75-OS/9 $60-UNIFLEX

FULL-SCREEN XBASIC PROGRAMS with cursor control
AVAILABLE FOR FLEX, UNIFLEX, AND MSDOS

DISPLAY GENERATOR/DOCUMENTOR	$50 w/source, $25 without
MAILING LIST SYSTEM	$100 w/source, $50 without
INVENTORY WITH MRP	$100 w/source, $50 without
TABULA RASA SPREADSHEET	$100 w/source, $50 without

DISK AND XBASIC UTILITY PROGRAM LIBRARY
$50-FLEX $30-UNIFLEX/MSDOS
edit disk sectors, sort directory, maintain master catalog, do disk sorts,
resequence some or all of BASIC program, xref BASIC program, etc.
non-FLEX versions include sort and resequence only

CMODEM TELECOMMUNICATIONS PROGRAM
$100-FLEX,OS/9,UNIFLEX,MS DOS,OS/9-68K,UNIX
OBJECT-ONLY versions: EACH $50
menu-driven with terminal mode, file transfer, MODEM7, XON-XOFF, etc.
for COCO and non-COCO; drives internal COCO modem port up to 2400 Baud

DISKETTES & SERVICES

5.25" DISKETTES
EACH 10-PACK $12.50-SSSD/SSDD/DSDD
American-made, guaranteed 100% quality, with Tyvek jackets, hub rings, and labels

ADDITIONAL SERVICES FOR THE COMPUTING COMMUNITY

CUSTOMIZED PROGRAMMING
we will customize any of the programs described in this advertisement or in our brochure for specialized customer use or to cover new processors; the charge for such customization depends upon the marketability of the modifications

CONTRACT PROGRAMMING
we will create new programs or modify existing programs on a contract basis, a service we have provided for over twenty years; the computers on which we have performed contract programming include most popular models of mainframes, including IBM, Burroughs, Univac, Honeywell, most popular models of minicomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800/1, 6809, Z80, 6502, 68000, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task

CONSULTING
we offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses

Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 30207
Telephone 404-483-4570 or 1717

We take orders at any time, but plan long discussions after 6, if possible.

Contact us about catalog, dealer, discounts, and services.
Most programs in source: give computer, OS, disk size.
25% off multiple purchases of same program on one order.
VISA and MASTER CARD accepted; US funds only, please.
Add GA sales tax (if in GA) and 5% shipping.

(UNI)FLEX tm Technical Systems Consultants; OS/9 Microware; COCO Tandy; MSDOS Microsoft

THE 6800-6809 BOOKS

..HEAR YE......HEAR

OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibbles **OS9 USER NOTES**

Information for the BEGINNER to the PRO, Regular or CoCo OS9

Using OS9
HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDARDS, Generating a New Bootstrap, Building a new System Disk, OS9 Users Group, etc.

Program interfacing to OS9
DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION, "SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

Programming Languages
Assembly Language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

Disks Include
No typing all the Source Listings in. Source Code and, where applicable, assembled or compiled Operating Programs. The Source and the Discussions in the Columns can be used "as is", or as a "Starting Point" for developing your OWN more powerful Programs. Programs sometimes use multiple Languages such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

BOOK $9.95

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)
Deluxe Binder - - - - - - - - - - - $5.50

All Source Listings on Disk

1-8" SS, SD Disk - - - - $14.95
2-5" SS, DD Disks - - - $24.95

FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVEC1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASC

NOTE: .C1,.C2, etc.=Chapter 1, Chapter 2, etc.

**Over 30 TEXT files included is ASM (assembler)-PASCAL-PIC (position independent code) TSC BASIC-C, etc.

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

Shipping & Handling $3.50 per Book, $2.50 per Disk set
Foreign Orders Add $4.50 Surface Mail or $7.00 Air Mail
If paying by check - Please allow 4-6 weeks delivery
• All Currency in U.S. Dollars

Continually Updated In 68 Micro Journal Monthly

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343

VISA

(615) 842-4601
Telex 5106006630

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware and Motorola
*68' Micro Journal is a trademark of Computer Publishing Inc.

OS-9™ SOFTWARE

L1 UTILITY PAK—Contains all programs formerly in Filter kits 1 & 2, and Hacker's kit 1 plus several additional programs. Complete "wild card" file operations, copies, moves, sorts, del, MACGEN shell command language compiler, Disassembler, Disk sector edit utility, new and improved editions, approx. 40 programs, increases your productivity. Most programs applicable for both level I & II 6809 OS-9. **$49.95** ($51.95)

Call or send Self Addressed Stamped Envelope for catalog of software for color Computer OS-9 and other OS-9 systems.

BOLD prices are CoCo OS-9 format disk, other formats (in parenthesis) specify format. All orders prepaid or COD, VISA and MasterCard accepted. Add $1.50 S&H on prepaid, COD actual charges added.

SS-50C
MEMORY LIQUIDATION SALE!
(While Supply Lasts)

1 MEGABYTE RAM BOARD

Full megabyte of ram with disable options to suit any SS-50 6809 system. High reliability, can replace static ram for fraction of the cost. $399 for 2 Mhz or $439 for 2.25 Mhz board assembled, tested and fully populated.

2 MEGABYTE RAM DISK BOARD

RD2 2 megabytes dedicated ram disk board for SS-50 systems. Four layer circuit board socketed for 2 **Megabytes!** Special sale price of **$399.00** includes only 256k of ram installed (you add the rest), includes OS-9 level I and II drivers for Ram disk, (note: you can reboot your system without losing ram-disk contents). (Add $6 shipping and insurance.)

Please call for answers to your technical questions concerning these products.

D.P. Johnson, 7655 S.W. Cedarcrest St.
Portland, OR 97223, (503) 244-8152
(For best service call between 9-11 am Pacific time.)

OS-9 is a trademark of Microwave and Motorola Inc.
MS-DOS is a trademark of Microsoft Inc.

COMPILER EVALUATION SERVICES

BY: Ron Anderson

The S.E. MEDIA Division of Computer Publishing Inc. is offering the following SUBSCRIBER SERVICE:

COMPILER COMPARISON AND EVALUATION REPORT

Due to the constant and rapid updating and enhancement of numerous compilers, and the different utility, appeal, speed, level of communication, memory usage, etc., of different compilers, the following services are now being offered with periodic updates.

This service, with updates, will allow you who are wary or confused by the various claims of compiler vendors, an opportunity to review comparisons, comments, benchmarks, etc., concerning the many different compilers on the market, for the 6809 microcomputer. Thus the savings could far offset the small cost of this service.

Many have purchased compilers and then discovered that the particular compiler purchased either is not the most efficient for their purposes of does not contain features necessary for their application. Thus the added expense of purchasing additional compiler(s) or not being able to fully utilize the advantages of high level language compilers becomes too expensive.

The following COMPILERS are reviewed initially, more will be reviewed, compared and benchmarked as they become available to the author:

PASCAL 'C' GSPL WHIMSICAL PL/9

Initial Subscription - $39.95
(includes 1 year updates)
Updates for 1 year - $14.50

S.E. MEDIA - C.P.I.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601

Call for Software:
68000, C, Basic09 Sculptor

We are receiving calls and letters from numerous sources, including users, business and others looking for OS-9 68000 software: applications, etc.

Many of you have developed software that with little change could be adapted for others. If you are interested in selling it, please let us know. There is a growing market out there now. Get in on the ground floor!

If you can use additional income and have something that might be of interest, call and talk to Larry or Don.

S.E. MEDIA Division - CPI

POB 849
Hixson, TN 37343

Telephone (615) 842-6809
Telex (510) 600-6630

Stop!

Get a 25 Mega-Byte Hard Disk practically FREE only 1¢

*Be Sure to Consider the SPECIAL MUSTANG-08/A 1¢ Sale on page 7

*This Is exactly one cent more than the price of the same system - with two floppies - for one cent more you get one floppy and a 25 MegaByte Hard Disk with the faster CPU board, additional serial ports and improved clock! Includes Professional OS-9™ Version 2 and the $500.00 C Compiler!

Remember - When it's over, IT'S OVER!

We don't know how long this very, very low price can be maintained, don't miss it!

Data-Comp Div. - CPI

6809<>68XXX
UniFLEX

X-TALK

A C-MODEM/Hardware Hookup

Exclusive for the MUSTANG-020 running UniFLEX, is a new transfer program and cable set from DATA-COMP (CPI). X-TALK consist of 2 disks and a special cable, this hook-up enables a 6809 SWTPC UniFLEX computer to port UniFLEX files directly to a 68XXX UniFLEX system.

This is the only currently available method to transfer files, text or otherwise, from a 6809 UniFLEX system to a 68000 UniFLEX system, that we have seen. A must if you want to recompile or cross assemble your old (and valuable) source files to run on a 68000 UniFLEX system. GIMIX users can directly transfer files between a 6809 GIMIX system and our MUSTANG-020 68020 system, or GIMIX 68020 system. All SWTPC users must use some sort of method other than direct disk transfer. The 6809 SWTPC UniFLEX disk format is not readable by most other 68000 type systems.

The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 DB25 connectors. A special SWPTC+ cable and software is also available, at the same price. Orders must specify which type SWTPC 6809 UniFLEX system they intend to transfer from or to.

The X-TALK software is furnished on two disks. One 8" disk containing the 6809 software and one 5" disk containing the 68XXX software. These programs are also complete MODEM programs and can be used as such, including X-on X-off, and all the other features you would expect from a full modem program.

X-TALK can be purchased with/without the special cables, however, this SPECIAL price is available only to registered MUSTANG-020 owners.

X-TALK, w/cable $ 99.95
X-TALK only 69.95
X-TALK w/source $149.95

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

Telephone 615 842-4601
Telex 510 600-6630

Note: Registered MUSTANG-020 owners must furnish system serial number in order to buy at these special low prices.

68 MICRO JOURNAL
Reader Service Disks

Disk- 1 Filesort, Minical, Minicopy,Minifms, **Lifetime, **Poetry, **Foodlist, **Diet.
Disk- 2 Diskedit w/ inst.& fixes, Prime, *Prmod, **Snoopy, **Football, **Hexpawn,**Lifetime.
Disk- 3 Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Disksave.
Disk- 4 Mailing Program, *Finddat, *Change, *Testdisk.
Disk- 5 *DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.
Disk- 6 **Purchase Order, Index (Disk file indx).
Disk- 7 Linking Loader, Rload, Harkness.
Disk- 8 Crtest, Lanpher (May 82).
Disk- 9 Datecopy, Diskfix9 (Aug 82).
Disk-10 Home Accounting (July 82).
Disk-11 Dissembler (June 84).
Disk-12 Modem68 (May 84).
Disk-13 *Initmf68, Testmf68, *Cleanup, *Dskalign, Help, Date.Txt.
Disk-14 *Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib
Disk-15 Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo).
Disk-16 Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc.
Disk-17 Match Utility, RATBAS, A Basic Preprocessor.
Disk-18 Parse.Mod, Size.Cmd (Sept. 85 Armstrong), CMDCODE, CMD.Txt (Sept. 85 Spray).
Disk-19 Clock, Date, Copy, Cat, PDEL.Asm & Doc., Errors.Sys, Do, Log.Asm & Doc.
Disk-20 UNIX Like Tools (July & Sept. 85 Taylor & Gilchrist). Dragon.C, Grep.C, LS.C, FDUMP.C.
Disk-21 Utilities & Games - Date, Life, Madness, Touch, Goblin, Starshot, & 15 more.
Disk-22 Read CPM & Non-FLEX Disks. Fraser May 1984.
Disk-23 ISAM, Indexed Sequential file Accessing Methods, Condon Nov. 1985. Extensible Table Driven. Language Recognition Utility, Anderson March1986.
Disk-24 68' Micro Journal Index of Articles & Bit Bucket Items from 1979 - 1985, John Current.
Disk-25 KERMIT for FLEX derived from the UNIX ver. Burg Feb. 1986. (2)-5" Disks or (1)-8" Disk.
Disk-26 Compacta UniBoard review, code & diagram, Burlison March '86.
Disk-27 ROTABIT.TXT, SUMSTEST.TXT, CONDATA.TXT, BADMEN.TXT.
Disk-28 CT-82 Emulator, bit mapped.
Disk-29 **Star Trek
Disk-30 Simple Winchester, Dec.'86 Green.
Disk-31 *** Read/Write MS/PC-DOS (SK*DOS)
Disk-32 Hier-UNIX Type upgrade - 68MJ 2/87

NOTE:
This is a reader service ONLY! No Warranty is offered or implied, they are as received by 68' Micro Journal, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other. Software is available to cross-assemble all.

* Denotes 6800 - ** Denotes BASIC
*** Denotes 68000 - 6809 no indicator.

Specify 8" disk $19.50
5" disk $16.95

Add: S/H - $3.50
Overseas add: $4.50 surface - $7.00 Air Mail, USA Dollars

68 MICRO JOURNAL
PO Box 849
Hixson, TN 37343
615 842-4600 - Telex 510 600-6630

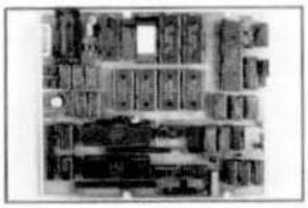
6809/68008 SINGLE BOARD COMPUTERS

The Peripheral Technology Family of Single Board Computers is a Low-Cost Group Which Ranges From an Entry Level 8-Bit Version to a Powerful 68008-Based Board. A Product is Available to Fit Almost Every User's Requirements.

PT69-5

- 6809 Processor/2MHZ Clock
- 4 RS-232 Serial Ports
- 2 8-Bit Parallel Ports
- 4K-16K EPROM/60K Ram
- Parallel Printer Interface
- DS/DD Controller for 35-80
- Track Drives Ranging From SS/SD-DS/DD
- Winchester Interface Port

Price: $315.00



PT69-3

- 6809 1 MHZ Processor
- 2 RS-232 Serial Ports
- 2 8-Bit Parallel Ports
- 4K EPROM/59K User Ram
- DS/DD Controller for 35-80 Track Drives Ranging From SS/SD-DS/DD

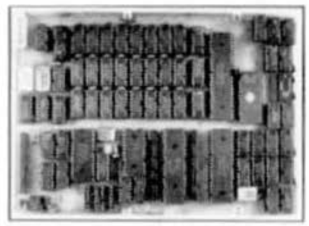
Price: $249.95

OS9 L1 For PT69 BOARDS: $200.00
SK*DOS: $ 49.95

PT68K-1A

- MC68008 12.5 MHZ Processor
- 768K RAM/64K EPROM
- 4-RS-232 Serial Ports
- Winchester Interface Port
- Floppy Disk Controller for 4 5¼" Drives
- 2 8-Bit Parallel Ports

BOARD: $499.00
with Professional OS9: $895.00
with SK*DOS: $595.00



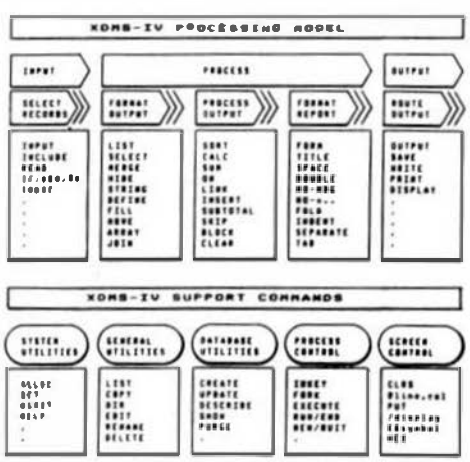
PERIPHERAL TECHNOLOGY
1480 Terrell Mill Road, Suite 870
Marietta, Georgia 30067
(404) 984-0742 Telex # 880584
VISA/MASTERCARD/CHECK/C.O.D.

*OS9 Is A Trademark Of Microware and Motorola.

Send For Catalogue For Complete Information On All Products

XDMS-IV

Data Management System



Save $100.00 - Limited Time
Regular $350.00 - ~~Now Only~~
$249.95

Technical telephone assistance: Tel 914-941-3552 (Evenings)
FLEX™ Tecnical Systems Consultants, SK*DOS™ STAR-KITS Corp.

FOR 6809 FLEX-SK-DOS(5/8")

Up to 32 groups/fields per record! Up to 12 character filed name! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced format! Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) oriented which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV!

IT'S EASY TO USE!

XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...). The possibilities are unlimited...

Visa & Master Card Excepted

Telephone: 615-842-4601 or Telex: 510 600-6630
Or Write: S.E. Media, 5900 Cassandra Smith Rd., Hixson, Tenn. 37343

GMX Inc. 1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4055

GMX MICRO 20 PRICE LIST

MICRO 20 (12.5 MHz) W/1 SAB...............$2565.00
MICRO 20 (16.67 MHz) W/1 SAB..............$2895.00
MICRO 20 (20 MHz) W/1 SAB.................$3295.00

OPTIONAL PARTS AND ACCESSORIES

68881RC12.................................$ 295.00
68881RC16.................................$ 395.00
MOTOROLA 68020 USERS MANUAL...............$ 18.00
MOTOROLA 68881 USERS MANUAL...............$ 18.00

SBC ACCESSORY PACKAGE (M20-AP)......$1690.00
The package includes a PC-style cabinet with a custom backpanel, a Xebec 1410A hard disk controller, a 25 Megabyte (unformatted) hard disk, a 5 1/4" DSDD 80 track floppy disk drive, a 150 watt power supply, cooling fan, panel mounted reset and abort switches and all necessary internal cabling
BACK PANEL PLATE (BPP-PC) For Above.......$ 44.00
2nd 5"80 FLOPPY & CABLES FOR M20-AP, ADD..$ 250.00
SECOND 25MB HARD DISK & CABLES, ADD.......$ 780.00
TO SUBSTITUTE 85MB HD FOR 25MB HD, ADD....$1800.00

I/O EXPANSION BOARDS

16 PORT SERIAL CARD ONLY (SBC-16S)........$335.00
The SBC-16S extends the I/O capabilities of the GMX Micro-20 68020 Single-board Computer by adding sixteen asynchronous serial I/O ports. By using two SBC-16S boards, a total of thirty-six serial ports are possible.

RS232 Adapter(SAB-25,SAB-9D or RJ-45).....$165.00
The board provides level-shifting between TTL level and standard RS-232 signal levels for up to 4 serial I/O ports.

60 LINE PARALLEL I/O CARD (SBC-60P).......$398.00
The GMX SBC-60P uses three 68230 Parallel Interface/Timers (PI/Ts) to provide up to forty-eight parallel I/O lines. The I/O lines are buffered in six groups of eight lines each, with separate buffer direction control for each group. Buffer direction can be fixed by hardware jumpers, or can be software programmable for bidirectional applications.

PROTOTYPING BOARD (SBC-WW)................$75.00
The SBC-WW provides a means of developing and testing custom I/O interface designs for the GMX Micro-20 68020 Single-board Computer. The board provides areas for both DIP (Dual Inline Package) and PGA (Pin Grid Array) devices, and a pre-wired memory area for up to 512K bytes of dynamic RAM.

I/O BUS ADAPTER (SBC-BA)..................$195.00
The SBC-BA provides an interface between the GMX Micro-20 68020 Single-board Computer and the Motorola Input/Output Channel (I/O bus). With the I/O bus, up to sixteen off-the-shelf or custom peripheral devices (I/O modules) can be connected to the GMX Micro-20.

ARCNET LAN board w/o Software (SBC-AN)....$475.00
The SBC-AN provides an interface between the GMX Micro-20 68020 Single-board Computer and the ARCNET modified token-passing Local Area Network (LAN) originally developed by Datapoint Corp. The ARCNET is a baseband network with a data transmission rate of 2.5 Megabits/second. The standard transmission media is a single 93 ohm RG-62/U coaxial cable. Fiber optic versions are available as an option.

OS9 LAN Software Drivers for SBC-AN.......$ 120.00

GMX MICRO 20 SOFTWARE

020 BUG UPDATE - PROMS & MANUAL...........$ 150.00

THESE 68020 OPERATING SYSTEMS ARE PRICED WHEN PURCHASED WITH THE MICRO-20. PLEASE ADD $150.00 IF PURCHASED LATER FOR THE UPDATED PROMS AND MANUALS.

OS9

OS9/68020 PROFESSIONAL PAK................$ 850.00
Includes O.S., "C", EDITOR, ASSEMBLER, DEBUGGER, development utilities, 68881 support.
Other Software for OS-9/68020
BASIC (included in PERSONAL PAK)..........$ 200.00
C COMPILER (included in PROFESSIONAL PAK).$ 500.00
PASCAL COMPILER...........................$ 500.00

UNIFLEX

UniFLEX (when ordered with Board).........$ 450.00
UniFLEX WITH REAL-TIME ENHANCEMENTS.......$1000.00
Other Software for UniFLEX
UniFLEX BASIC W/PRECOMPILER...............$ 300.00
UniFLEX C COMPILER........................$ 350.00
UniFLEX COBOL COMPILER....................$ 750.00
UniFLEX SCREEN EDITOR.....................$ 150.00
UniFLEX TEXT PROCESSOR....................$ 200.00
UniFLEX SORT/MERGE PACKAGE................$ 200.00
UniFLEX VSAM MODULE.......................$ 100.00
UniFLEX UTILITIES PACKAGE I...............$ 200.00
UniFLEX PARTIAL SOURCE LICENSE............$1000.00

GMX EXCLUSIVE VERSIONS,CUSTOMIZED FOR THE MICRO-20, OF THE BELOW LANGUAGES AND SOFTWARE ARE ALSO AVAILABLE FROM GMX.

ABSOFT FORTRAN (UniFLEX)..................$1500.00

SCULPTOR (specify UniFLEX or OS9).........$ 995.00

FORTH (OS9)...............................$ 595.00

DYNACALC (specify UniFLEX or OS9).........$ 300.00

GMX DOES NOT GUARANTEE PERFORMANCE OF ANY GMX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURES PRODUCT.

ALL PRICES ARE F.O.B. CHICAGO

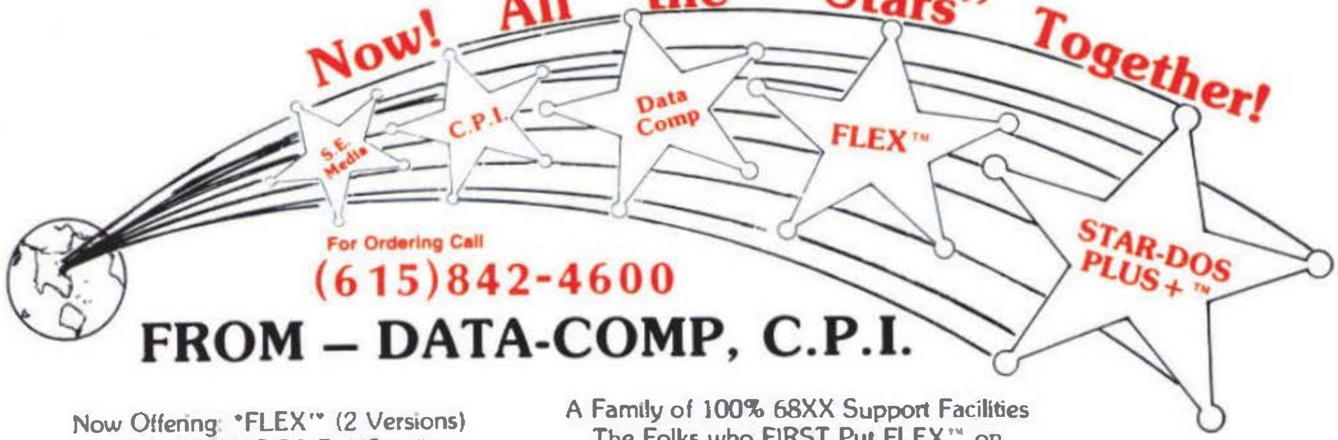
GMX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account number 73-32033.

CONTACT GMX FOR MORE INFORMATION ON THE ABOVE PRODUCTS

GMX STILL SELLS GIMIX SS0 BUS SYSTEMS, BOARDS & PARTS. CONTACT GMX FOR COMPLETE PRICE LIST.

Now! All the "Stars" Together!



For Ordering Call

(615)842-4600

FROM – DATA-COMP, C.P.I.

Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS + ™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00 **Only $79.95**

STAR-DOS PLUS +
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.95

FLEX-CoCo Jr.
without TSC
Editor & Assembler
$49.95

PLUS

ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Editor
Reg $50.00
NOW $35.00

TSC Assembler
Reg $50.00
NOW $35.00

CoCo Disk Drive Systems

2 THINLINE DOUBLE SIDED DOUBLE DENSITY DISK DRIVES SYSTEM WITH POWER SUPPLY, CABINET, DISK DRIVE CABLE, J&M NEW DISK CONTROLLER JFD-CP WITH J-DOS, RS-DOS OPERATING SYSTEMS.	$469.95

* Specify What CONTROLLER You Want J&M, or RADIO SHACK

THINLINE DOUBLE SIDED DOUBLE DENSITY 40 TRACKS	$129.95

Verbatim Diskettes

Single Sided Double Density	$ 24.00
Double Sided Double Density	$ 24.00

Controllers

J&M JFD-CP WITH J-DOS	$139.95
WITH J-DOS, RS-DOS	$159.95
RADIO SHACK 1.1	$134.95
RADIO SHACK Disk CONTROLLER 1.1	$134.95

Disk Drive Cables

Cable for One Drive	$ 19.95
Cable for Two Drives	$ 24.95

MISC

64K UPGRADE FOR C,D,E,F, AND COCO II	$ 29.95
RADIO SHACK BASIC 1.2	$ 24.95
RADIO SHACK DISK BASIC 1.1	$ 24.95
DISK DRIVE CABINET FOR A SINGLE DRIVE	$ 49.95
DISK DRIVE CABINET FOR TWO THINLINE DRIVES	$ 69.95

PRINTERS

EPSON LX-80	$289.95
EPSON MX-70	$125.95
EPSON MX-100	$495.95

ACCESSORIES FOR EPSON

8148 2K SERIAL BOARD	$ 89.95
8149 32K EXPAND TO 128K	$169.95
EPSON MX-RX-80 RIBBONS	$ 7.95
EPSON LX-80 RIBBONS	$ 5.95
TRACTOR UNITS FOR LX-80	$ 39.95
CABLES & OTHER INTERFACES CALL FOR PRICING	

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

SHIPPING
USA ADD 2%
FOREIGN ADD 5%
MIN. $2.50

(615)842-4600
For Ordering
Telex 5106006630

Introducing ....

S - 50 BUS / 68XX

Board and/or Computer
Terminals-CRTs-Printers
Disk Drives-etc.

REPAIRS

This

Not This

NOW AVAILABLE TO ALL S50/68XX USERS

The Data-Comp Division of CPI is proud to announce the availability of their service department facilities to '**ALL**' S50 Bus and 68XX users. Including all brands, SWTPC - GIMIX - SSB - HELIX and others, including the single board computers. *Please note that kit-built components are a special case, and will be handled on an individual basis, if accepted.

1. If you require service, the first thing you need to do is call the number below and describe your problem and *confirm a Data-Comp service & shipping number!* This is very important, Data-Comp will not accept or repair items not displaying this number! Also we cannot advise or help you troubleshoot on the telephone, we can give you a shipping number, but **NO** advice! Sorry!

2. All service shipments must include both a minimum $40.00 estimate/repair charge and pre-paid return shipping charges (should be same amount you pay to ship to Data-Comp).

3. If you desire a telephone estimate after your repair item is received, include an additional $5.00 to cover long distance charges. Otherwise an estimate will be mailed to you, if you requested an estimate. Estimates *must be requested.* Mailed estimates slow down the process considerably. However, if repairs are not desired, after the estimate is given, the $40.00 shall constitute the estimate charge, and the item(s) will be returned unrepaired providing sufficient return shipping charges were included with the item to be serviced. Please note that estimates are given in dollar amounts only.

4. Data-Comp service is the oldest and most experienced general S50/68XX service department in the world. We have over $100,000.00 in parts in stock. We have the most complete set of service documents for the various S50/68XX systems of anyone - **YET, WE DO NOT HAVE EVERYTHING!** But we sure have more than anyone else. We repair about 90% of all items we receive. Call for additional information or shipping instructions.

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

(615)842-4607
Telex 5106006630

VISA